

**AFRL-IF-RS-TR-2003-178**  
**Final Technical Report**  
**August 2003**



# **ULTRA-HIGH CAPACITY NETWORKING ENABLED BY OPTICAL TECHNOLOGIES**

**Drexel University**

**Sponsored by**  
**Defense Advanced Research Projects Agency**  
**DARPA Order No. J181**


*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

**AIR FORCE RESEARCH LABORATORY**  
**INFORMATION DIRECTORATE**  
**ROME RESEARCH SITE**  
**ROME, NEW YORK**

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-178 has been reviewed and is approved for publication.

APPROVED:   
PAUL SIERAK  
Project Engineer

FOR THE DIRECTOR: 

WARREN H. DEBANY JR., Technical Advisor  
Information Grid Division  
Information Directorate

<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved</i> <b>OMB No. 074-0188</b>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
<b>1. AGENCY USE ONLY (Leave blank)</b>		<b>2. REPORT DATE</b> AUGUST 2003	<b>3. REPORT TYPE AND DATES COVERED</b> Final Nov 99 – Sep 02	
<b>4. TITLE AND SUBTITLE</b> ULTRA-HIGH CAPACITY NETWORKING ENABLED BY OPTICAL TECHNOLOGIES			<b>5. FUNDING NUMBERS</b> C - F30602-00-2-0501 PE - 62110E PR - J181 TA - 23 WU - 01	
<b>6. AUTHOR(S)</b> Stewart Personick				
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Drexel University 3141 Chestnut Street Philadelphia Pennsylvania 19104			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>  N/A	
<b>9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> Defense Advanced Research Projects Agency AFRL/IFGA 3701 North Fairfax Drive 525 Brooks Road Arlington Virginia 22203-1714 Rome New York 13441-4505			<b>10. SPONSORING / MONITORING AGENCY REPORT NUMBER</b>  AFRL-IF-RS-TR-2003-178	
<b>11. SUPPLEMENTARY NOTES</b>  AFRL Project Engineer: Paul Sierak/IFGA/(315) 330-7346/ Paul.Sierak@rl.af.mil				
<b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b> APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.				<b>12b. DISTRIBUTION CODE</b>
<b>13. ABSTRACT (Maximum 200 Words)</b> Project Pegasus, as this project came to be called, consisted of collaborative work in the following four areas: exploration of a novel internet protocol packet router using waveguide grating router devices; research into next generation internet backbone networks that are scalable by 1000 time the aggregate capacity of current internet backbone networks; development of applications requiring very high speed networking that enable leading edge research; and finally the design and demonstration of an all optical regenerator based upon terahertz optical asymmetric demultiplexes.				
<b>14. SUBJECT TERMS</b> IP Optical Packet Router, Terhertz Optical Asymmetric Demultiplier, TOAD, Optical Backbone Network				<b>15. NUMBER OF PAGES</b> 411
				<b>16. PRICE CODE</b>
<b>17. SECURITY CLASSIFICATION OF REPORT</b>  UNCLASSIFIED	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b>  UNCLASSIFIED	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b>  UNCLASSIFIED	<b>20. LIMITATION OF ABSTRACT</b>  UL	

## Table of Contents

<b>A. EXECUTIVE SUMMARY / INTRODUCTION .....</b>	<b>1</b>
<b>B. ULTRA HIGH CAPACITY IP ROUTER .....</b>	<b>2</b>
<b>b.1 Ultra High Capacity IP Router (Martin Zirngibl, et. al.) .....</b>	<b>2</b>
Overview .....	2
Early and intermediate results: .....	3
Final results: .....	4
<b>C. ARCHITECTURE .....</b>	<b>5</b>
<b>c.1 Network Congestion and QoS Management Strategies (Stewart Personick, et. al.).....</b>	<b>5</b>
<b>c.2 QoS Architectures and Mechanisms (Harish Sethu, et. al.) .....</b>	<b>8</b>
Conceptual advances .....	8
Algorithmic advances .....	8
Engineering implications.....	8
Results .....	9
Insights derived and lessons learned .....	12
<b>c.3 Embedded Network Processing (Jonathan Smith, et. al.) .....</b>	<b>13</b>
Background.....	13
ABIDE.....	14
PIGLET .....	14
Summary .....	15
<b>c.4 IP over WDM (Mohamed Ali, et. al.) .....</b>	<b>15</b>
Background.....	15
Results .....	16
Lessons learned .....	19
<b>c.5 Network Traffic Modeling (Stewart Personick, et. al.).....</b>	<b>20</b>
Background.....	20
Progress .....	21
<b>c.6 Wavelength Agility and Optical Networks (Bahram Nabet, Janet Jackel, et. al.).....</b>	<b>21</b>
Wavelength-selective optical detector (B. Nabet, et. al) .....	22
Wavelength agility in optical network elements ( Janet Jackel, et. al.).....	22
Background.....	22
Results .....	23
Lessons learned .....	32
<b>D. BIO-INFORMATICS AND BIO-COMPLEXITY APPLICATIONS .....</b>	<b>34</b>
<b>d.1 Distributed Neuroanatomical Database (J. “Yoni” Nissanov, et. al. and Susan Davidson, et. al.) .....</b>	<b>34</b>



<b>d.2 Cellular Observatory (J. Yasha Kresh, Banu Onaral, et. al.).....</b>	<b>35</b>
Part I - Telemic Software Development.....	36
Part II—Cellular Network Dynamics .....	38
Enhancements and Improvements .....	41
Telemicroscopy Software version 2.0 .....	43
New Horizons.....	45
Live Cell Microscopy- Summary of Results and Lessons Learned.....	46
<b>d.3 Gbps Infrastructure (Stewart Personick, et. al.; Dom Imbesi, et. al.) .....</b>	<b>47</b>
<b>d.4 Global Digital Mapping (Somerset Geographics: Brian Schmolt) .....</b>	<b>48</b>
<b>d.5 MONET East Ring (Lucent) .....</b>	<b>51</b>
<b>E. ALL-OPTICAL NETWORKING DEVICES AND SUBSYSTEMS .....</b>	<b>52</b>
<b>e.1 All-Optical Networking Devices and Subsystems (Paul Prucnal, et. al.) .....</b>	<b>52</b>
Optical pulse width management / format converter.....	52
Experiments to test a method for reducing data timing jitter, for use in our proposed 3R regenerator.....	52
All-Optical Clock Recovery .....	57
<b>REFERENCES (GROUPED BY REPORT SECTION) .....</b>	<b>62</b>
<b>APPENDIX (REPRESENTATIVE PUBLISHED PAPERS).....</b>	<b>69</b>

## List of Figures

<b>Figure 1: Lucent WGR-based Router</b> .....	2
<b>Figure 2: Combined Deterministic and Statistical Switching</b> .....	7
<b>Figure 3: Combined Deterministic and Statistical Switching Node</b> .....	7
<b>Figure 4: Time response to femtosecond laser with wavelength of 850 nm</b> .....	22
<b>Figure 5a: A waveband add/drop that is constructed from wide passband Bragg gratings</b> .....	25
<b>Figure 5b: Measured transmission and reflection for a fiber Bragg grating with a performance appropriate for a waveband add/drop</b> .....	26
<b>Figure 6: Waveband add/drop based on a dynamic band blocker</b> .....	27
<b>Figure 7: Schematic picture of liquid crystal switch used as dynamic beam blocker or band add/drop. Input and output spectra are shown.</b> .....	28
<b>Figure 8: Band add/drop based on a the liquid crystal switch</b> .....	29
<b>Figure 9: Comparing the passbands of two types of band selectors with the spectrum of a four-wavelength source used to fill the band.</b> .....	31
<b>Figure 10: Power in each of two orthogonal polarizations for wavelengths within a band and in different bands, after 76 km of fiber in a laboratory, where all wavelengths entered the fiber in the same polarization.</b> .....	31
<b>Figure 11: Cellular Observatory Remote-Access Graphical User Interface</b> .....	38
<b>Figure 12: Formation of robots changing shapes</b> .....	40
<b>Figure 13: Improved GUI</b> .....	44
<b>Figure 14: Live adult human (mesenchymal) stem cells co-cultured with human cardiac myocytes</b> .....	47
<b>Figure 15: TOAD Schematic</b> .....	52
<b>Figure 16: Experimental Setup for Timing Jitter Reduction</b> .....	53
<b>Figure 17: Input to the TOAD, with added timing jitter</b> .....	54
<b>Figure 18: The output of the TOAD, when we inject a third wavelength of CW light into the TOAD</b> .....	54
<b>Figure 19: The output after the TOAD, with the CW bias light level set at: +7dBm</b> .....	55
<b>Figure 20: BER vs. CW power</b> .....	56
<b>Figure 21: BER vs. SOA current</b> .....	56
<b>Figure 22: All-optical 3R Regenerator</b> .....	57
<b>Figure 23: Experimental structure of mode-locked figure-eight laser using a TOAD</b> .....	58
<b>Figure 24: Waveform and spectrum of the mode-locked output pulses</b> .....	59
<b>Figure 25: Pulse width, spectrum, and time-bandwidth product vs. the Offset of the SOA in the TOAD [10 ps input pulses]</b> .....	60
<b>Figure 26: Pulse width, spectrum, and time-bandwidth product vs. the Offset of the SOA in the TOAD [5 ps input pulses]</b> .....	61
<b>Table 1: Summary of Wavelength Conversion Alternatives</b> .....	30
<b>Table 2: US Coverage by DOQQ</b> .....	50

## Acknowledgement

The researchers who contributed to Project Pegasus wish to thank the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory for their support of this research. In particular, we wish to thank Dr. Mari Maeda who led the *Next Generation Internet* initiative at DARPA, and who sponsored and guided this research.

The researchers also wish to thank their respective universities and firms for providing their support.

## A. Executive Summary / Introduction

Project Pegasus (*Ultra-High Capacity Networking Enabled by Optical Technologies*) is a research program that was conducted from September 1999 to September 2002. It was sponsored by DARPA under the DARPA *Next Generation Internet* initiative, and funded under a contract to Drexel University from the US Air Force Research Laboratory.

There are four (4) major components of Project Pegasus:

The construction and demonstration of proof-of-concept-viability of a novel approach for constructing ultra-high-capacity Internet protocol, IP, packet routers using a passive optical device called a waveguide grating router, WGR.

The exploration and creation of technologies, methodologies, architectures, and insights for the implementation of next generation Internet backbone networks that will: scale to 1000 times the aggregate capacity of today's Internet backbone networks; be able to provide quality-of-service (specified limits on delay, loss, and delay jitter) guarantees in an efficient manner; be technically viable and cost effective, and that will be viable from a network management perspective,

The design, and implementation of compelling prototype applications of very high speed networking (>100 Mbps) by subject matter experts in fields outside of the domains of electrical engineering and computer science so as to demonstrate the ability of very high speed networking to enable leading edge research in other fields.

The design and demonstration of applications of Terahertz Optical Asymmetric Demultiplexer, TOAD, devices in the implementation of an all-optical digital regenerator that includes: reshaping, re-clocking, and regeneration, all in the optical domain.

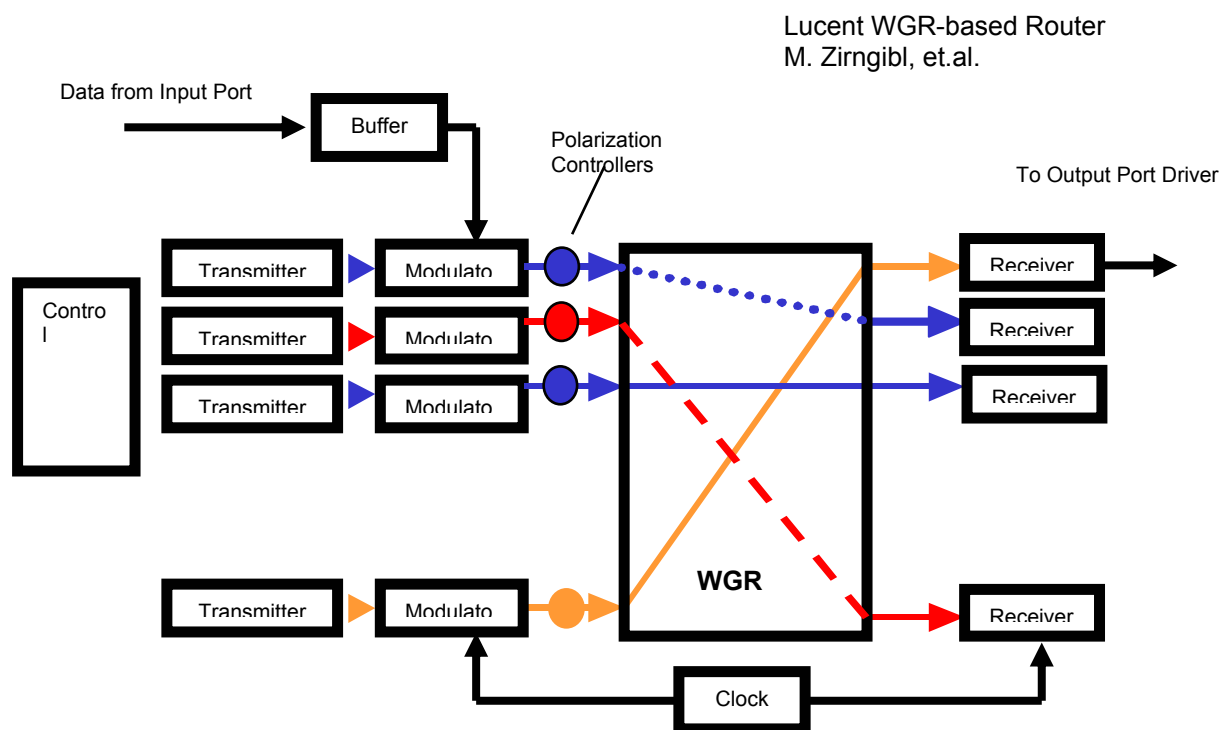
The WGR-based ultra-high-capacity router research was conducted by Martin Zirngibl, et. al., of Lucent Technologies. The technologies, methodologies, architectures, and insights research was conducted by: Stewart Personick, Harish Sethu, Bahram Nabet, et. al., of Drexel University; Jonathan Smith, et. al., of the University of Pennsylvania; Mohamed Ali, et. al., of The City College of the City University of New York; Janet Jackel, et. al, of Telcordia Technologies, and Al Schneider, of Verizon. The prototype application design and implementation research was conducted by: Banu Onaral, J. Yasha Kresh, Jonathan Nissanov, et. al., of Drexel University and MCP Hahnemann University (now part of Drexel University); Susan Davidson, et. al, of the University of Pennsylvania; Brian Schmult, of Somerset Geographics; and Dom Imbesi, et. al. of Lucent Technologies. The all-optical regenerator research was conducted by Paul Prucnal, et. al., of Princeton University.

## B. Ultra High Capacity IP Router

### b.1 Ultra High Capacity IP Router (Martin Zirngibl, et. al.)

#### Overview

The objective of this research is to demonstrate proof-of-concept viability of an ultra high throughput ( $>1$  billion packets per second) packet router, with a large number of input and output ports ( $>128 \times 128$ ), that employs a core packet routing fabric that consists of: an all-passive waveguide grating router, WGR, wavelength-tunable transmitters with a wavelength tuning time of a small fraction of the duration of a single packet ( $<100$  ns), and optical receivers capable of adjusting themselves to incoming packets having different power levels and a range of relative delays (see figure 1).



**Figure 1: Lucent WGR-based Router**

The prospective benefit of this router over conventional routers with electronic core switching fabrics is the ability to interconnect a large number of input and output modules (that can be physically separated) via optical fibers terminating on a completely passive core interconnect fabric. This eliminates two very challenging physical design problems of conventional approaches: rapidly increasing levels of heat production and crosstalk as the numbers of ports and the port data rates increase.

The prospective benefit of this router over proposed “all-optical” router concepts is that optical components are not used to perform packet buffering and packet alignment functions. It is an extremely difficult, unsolved challenge to design a practical all-optical buffer, with sufficient buffer delay and packet capacity. The known applications for a packet router that does not have the ability to buffer incoming packets are very limited.

The technical challenges for concept viability are:

- a. Designing and demonstrating reproducible and reliable waveguide grating routers, WGRs, with a large number of input and output ports ( $>128 \times 128$ ), low crosstalk between the optical signals passing through the WGR, as measured at the output ports of the router ( $< -25$  dB), low insertion loss ( $<10$  dB), low path-dependent insertion loss variability ( $< 3$  dB), low polarization-dependent loss ( $<1$  dB), low polarization-dependent dispersion ( $< 1$  ps).
- b. Designing tunable laser transmitter modules with a sufficiently wide tuning range ( $>256 \times$  the wavelength spacing associated with the WGR), a sufficiently fast tuning time ( $<100$  ns), a stable relationship between the wavelength emitted and the wavelength control signal values, and that can be modulated at very high data rates ( $> 40$  Gbps)
- c. Designing optical receivers that can quickly adjust ( $< 10$  ns) to the optical power level and relative delay of an incoming optical packet, and that operate at very high data rates ( $>40$  Gbps)
- d. Designing a router control architecture and associated routing algorithms that scales in a computationally tractable manner as the number of input and output ports increases.

The research results reported here were produced over an approximately one-year period during which this research was partially funded under this project (Pegasus). This research has continued, at Lucent Technologies, subsequent to the end of the period in which partial funding was provided under this project.

### **Early and intermediate results:**

A  $100 \times 200$  WGR was successfully assembled. The measured insertion losses ranged between 7 and 9 dB. In this first  $100 \times 200$  WGR the adjacent cross-talk was  $-15 / -20$  dB while the average cross-talk was less than  $-30$  dB. The polarization dependence has been completely removed from smaller test routers present on the same wafer. This shows that even in the larger device this impairment can be corrected.

The design of a new  $80 \times 80$  WGR was completed. Improvements over the previous design are: polarization independence, reduced next-neighbor cross-talk ( $-30$  to  $-35$  dB) and uniform channel spacing.

100 channels were calibrated on the first tunable-laser transmitter board (i.e., the relationship of the wavelength to the applied wavelength-control current, for 100 discrete wavelength-current pairs was determined). Two more tunable-laser transmitter boards were built, and 80 channels on a second tunable-laser transmitter board have been calibrated. The tuning range for the second tunable laser transmitter is 95 nm. A wavelength locker is being developed to perform wavelength stabilization.

Of three circuit boards designed to perform the buffering and scheduling, the first one has been built and tested. 10 Gb/s data patterns have been generated with this board.

A systems experiment has been performed to demonstrate optical switching of data packets. The experiment used one transmitter and the first 80x80 WGR. A total throughput of 1 Tb/s could be projected with the used components. The results were presented as a post-deadline paper at ECOC 2000 [1].

### **Final results:**

#### **Progress on the Waveguide Grating Router (WGR)**

Two different kinds of 40x40 WGRs were fabricated, both including integrated wavelength-monitoring capabilities. One type of device with non-cyclic response (fully packaged) shows insertion losses smaller than 3.5 dB while the version with cyclic response has approximately 8 dB of insertion loss. Polarization-dependent loss, PDL, is less than 0.3 dB and polarization dispersion, PD, is negligible. The crosstalk has been reduced to unprecedented low levels: adjacent cross-talk <-32 dB and background cross-talk level close to -50 dB.

#### **Progress on the wavelength-tunable laser transmitter:**

An improved drive circuit, with current amplification, was designed and implemented on the laser-driver circuit boards. The switching speed, for all before-after channel combinations, has been measured with this board. The laser switching performance was greatly improved over the previous results. Another version of the circuit board has been redesigned with more advanced digital electronics that allow on-board programming of the calibration lookup table. A wavelength locker is being developed to perform wavelength stabilization and channel calibration.

#### **Progress on the receiver:**

Various burst-mode clock and data recovery schemes are being evaluated and tested. Optical packet switching at 10 Gb/s has been performed to test the fast clock recovery. The recovery time was between 40 and 200 bits. Clock recovery modules at 40Gb/s are currently being developed.

#### **Progress on the scheduler and interface part:**

Work on an ASIC that serves as a scheduler and as a low speed interface to the fabric was started. The top-level architecture of this ASIC has been designed and implementation details were investigated. 40 Gb/s electronics have been designed. A first version of 40 Gbps transmitter electronics is operational, and was tested. The receiver electronics were also tested.

#### **Progress on the architecture:**

The top-level line card architecture has been finalized and details were refined. The scheduler architecture has been further refined, and performance and stability of the developed algorithms was tested by simulations. Concepts have been developed for extended architectures with increased throughput of hundreds of Tb/s.

## C. Architecture

This research is focused on the exploration of fundamental changes to the architecture and design methodologies for next generation Internets that might be required or desirable to meet the challenges of: scaling the aggregate Internet backbone capacity by a factor of 1000, and providing quality of service guarantees (packet delay, packet loss probability, maximum outage duration, delay jitter, etc.) for heterogeneous Internet traffic [9].

### **c.1 Network Congestion and QoS Management Strategies (Stewart Personick, et. al.)**

This research is focused on the challenges of scaling routers to large aggregate throughputs (>1 billion packets per second) and providing varying levels of packet routing quality of service (buffering delay and packet loss) while maintaining reasonably efficient utilization of the most expensive router resources.

We have taken an “engineering” approach by focusing on what we want to achieve in terms of router performance (as perceived by the users and operators of networks that employ routers) in next generation backbone networks. We then looked for new approaches to routing that recognize the different engineering tradeoffs that may be appropriate to next generation networks.

We have attempted to find new ways of using optical components in combination with electronic components that leverage the strengths of each. We have attempted to move beyond existing “all optical switching” concepts that are limited by their lack of a viable solution to the packet buffering requirement of real networks.

We have focused a great deal of time and thought on the issue of: whether or not there is a network architecture that can gainfully employ bufferless routers. We have not found any network design approaches that can do so, without pushing the buffer problem to the network elements that are feeding their inputs. In the case of the waveguide grating router (see section B of this report), pushing the buffering problem from the core switching fabric back to the co-located port modules is a good example of a router design approach that combines the strengths of optical and electronic components. However, the complete router in that case is the combination of the optical core switching fabric and the electronic port modules. The team designing the WGR-based router knows that one cannot define a router as just the optical core.

After much thought and analysis, we have focused on a concept for combining deterministic packet switching (leveraging time-division-switching concepts and theory) with conventional store-and-forward statistical packet switching. This has numerous engineering advantages, such as a single, combined hardware routing platform; the ability to set up virtual circuits with fixed delays; the ability to provide priority and un-interrupted service to the virtual circuit traffic via the store-and-forward traffic; and the ability to utilize unneeded virtual circuit capacity on the network links for store-and-forward traffic.



The concept is based on the use of fixed length time slots (to hold packets or packet fragments) on links between switching nodes; where some number,  $N$ , of time slots fall within a repeating cycle of time slots. Each time slot can be allocated to deterministically switched (circuit-switched) packet traffic, or conventional statistically-switched packet traffic. Deterministic traffic passes through a series of time slot interchangers, TSIs, and time divided switching fabrics. as in conventional time-division-switching. Time slot interchangers are required to resolve contention that occurs when two incoming packets, destined for the same output port, arrive in the same time slot on two different input ports. Store-and-forward statistically switched traffic passes through a conventional router fabric, where it is buffered and queued for access to an available slot on its desired outgoing port. Figure 2 and 3 conceptually portray the combination of deterministic and statistical switching.

An overview of the motivation for using combined deterministic packet (or burst) switching and statistical packet (or burst) switching in next generation Internet backbone networks, and the potential technical advantages of this approach in moving toward routers/switches with higher throughput capabilities, lower/deterministic latency, and improved quality of service, QoS, as well as the improved feasibility of employing optical buffers to perform the required time-slot-interchange function, was presented at the 2001 Topical Meeting on Photonics in Switching, Monterey CA , June 13-15, 2001 [10].

A preliminary patent application was filed with the U.S. Patent and Trademarks Office, which disclosed the overall concept and enabling implementation details. The patent application specification (following up on the preliminary patent application) was completed as of September 30, 2001. The patent application was filed with the U.S. Patent and Trademarks Office the first week of October 2001.

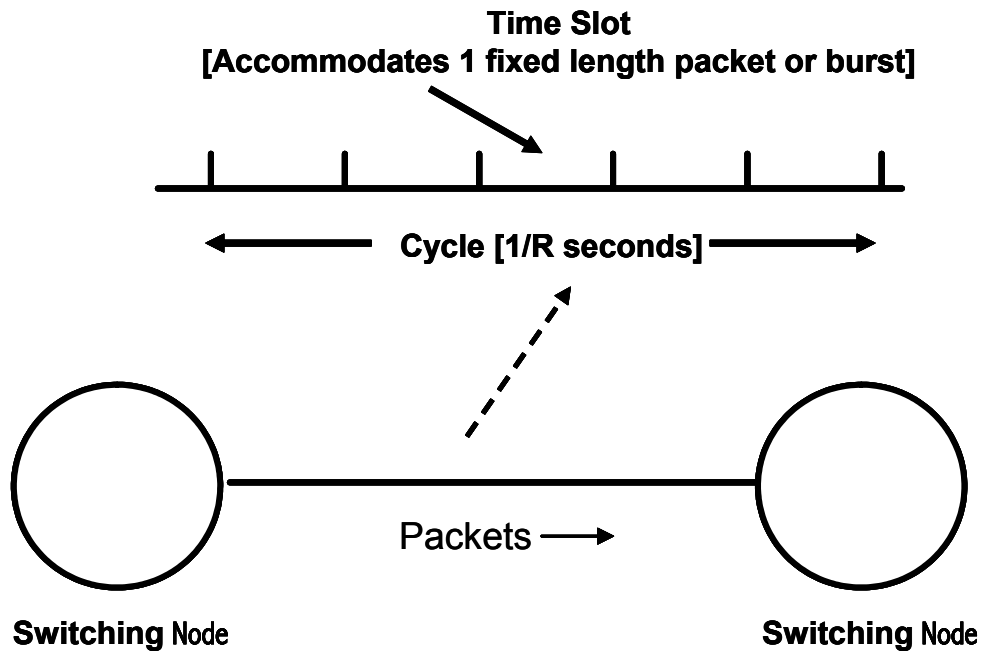


Figure 2: Combined Deterministic and Statistical Switching

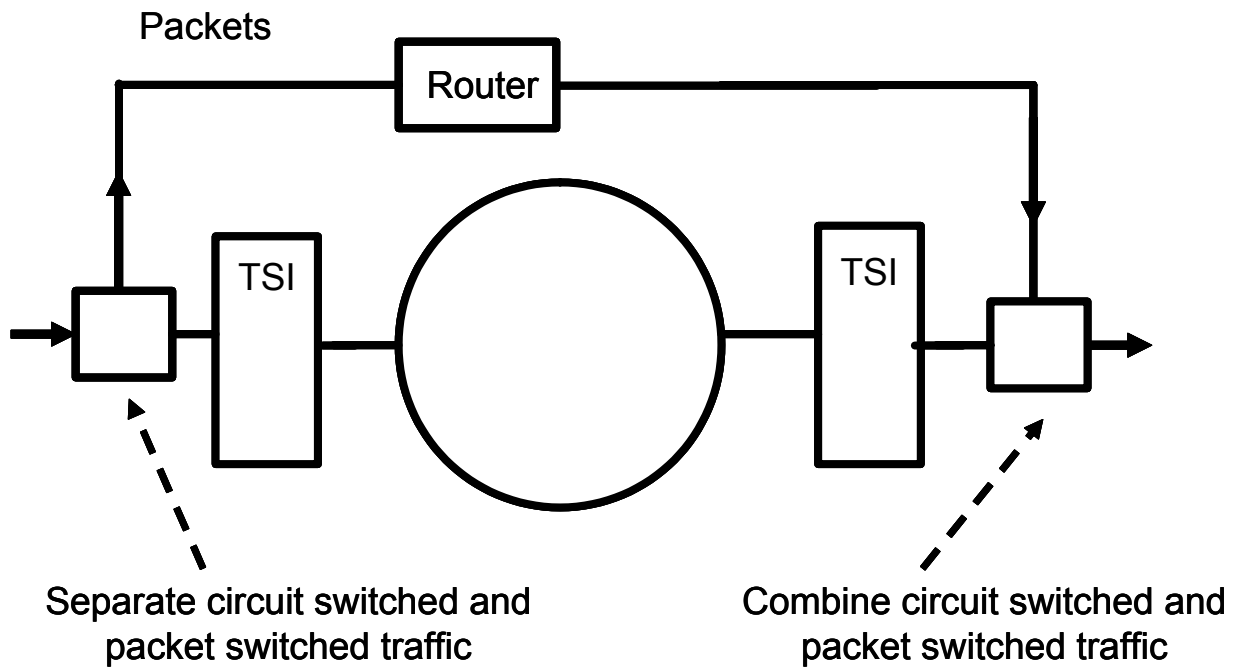


Figure 3: Combined Deterministic and Statistical Switching Node

## **c.2 QoS Architectures and Mechanisms (Harish Sethu, et. al.)**

The goal of this research is to facilitate the evolution of the future Internet as a scalable high-performance infrastructure for multimedia communications with differentiated levels of service. To this end, we seek novel advances in three complementary and interdependent areas: conceptual advances that allow crisp definitions of QoS parameters and requirements; algorithmic advances that improve upon earlier methods of achieving QoS; and finally the implications of these on the design of routers, switches and system architectures.

### **Conceptual advances**

An important conceptual advance is in developing an understanding of the true requirements of multimedia traffic under new emerging Internet service models such as Integrated and Differentiated services. While several service models have been defined by the Internet Engineering Task Force, IETF, mechanisms, such as scheduling disciplines and traffic management algorithms to support these services, do not yet exist. However, devising new mechanisms requires non-trivial conceptual advances in actually determining the requirements to be placed on these mechanisms. Our work provides a set of these requirements for a few popular service classes such as the controlled load service in Integrated Service, IntServ, and the expedited forwarding – per hop behavior, EF-PHB, in Differentiated Services, DiffServ.

A second conceptual advance is with regard to fairness, since the most basic guarantee desired by any flow is fairness in the treatment of its packets. Fairness has been well-studied in the context of sharing bandwidth on a link. However, flows share more resources than just bandwidth on a link, and therefore, there is a need to make a conceptual leap and define a more over-arching principle of network behavior that includes guidelines for sharing buffers or CPU cycles. Our work provides this conceptual advance, and presents a theory of fairness that is significantly more generalized than the classic principles commonly cited today. In particular, we define what is fair in buffer allocation, and also what is fair in the distribution of CPU cycles in routers and switches.

### **Algorithmic advances**

Scheduling disciplines are critical to the quality of service attained by multimedia traffic. We developed new scheduling algorithms that improve upon the following: fairness, latency, scalability and efficiency. Properties of the new algorithms have been analytically proven and established, with further verification by simulation. The goal of our work is to come up with new algorithms that are efficient and scalable, and therefore, our work restricts itself to the design of algorithms with a work complexity of  $O(1)$  with respect to the number of flows. New scheduling algorithms, that have been devised, complement our conceptual advances, and together with the study of engineering implications, lead toward a complete solution.

### **Engineering implications**

This relates to the hardware, software or architectural means by which new algorithms and conceptual advances will eventually be realized. The goal of this work is to provide complete solutions to the quality of service issues tackled, and therefore, our goal is to ensure the engineering feasibility of the new advances. To this end, we perform research on the relevant

engineering implications and study the practicality of the new algorithms and architectural strategies developed as part of our work. For example, in our research on a new scheduling discipline for the controlled load service in the Internet, we also analyzed the hardware/software implementation scenarios and the associated complexities.

## **Results**

We developed a set of economically based principles and associated mechanisms that routers should use to handle traffic with a heterogeneous set of QoS requirements. There are at least two economic principles that one can use in determining the equivalence between the diverse requirements of multiple flows. When the resource availability is adequate, one could try to equalize the satisfactions achieved by flows at any given instant of time (through maximizing the Gini index, a popular measure of inequality in economics). When the resources available are inadequate, one could maximize the sum of the satisfactions of all the flows, thus sacrificing a few heavy consumers to the benefit of many light consumers. In the case of bandwidth allocation in the presence of adequate bandwidth, we are currently devising a practical algorithm that can maximize the Gini index and thus achieve almost perfect fairness and very tight delay bounds. The algorithm, and our principle allow extrapolation into a variety of resource allocation problems besides bandwidth allocation. Our work is ongoing on applying the same principle for allocations based on deterministic delay requirements. Our work is also ongoing on the policies to use in the absence of adequate resources.

We developed a router/switch architecture that is scalable as the bandwidth of the link grows to something significantly more than the software or the internal memory access speeds can sustain. For example, with the advent of optical networking, we are presented with a situation where the memory speed, in switches, is significantly lower than the link speeds. I.e., the number of bits that have to be moved in and out of memory per unit time is much larger than the memory read/write cycle speed. A solution to this problem is to implement wide memories (large number of bits read/written per operation); however, this introduces the problem of memory fragmentation since most packet lengths may be smaller than the width of the memory itself. A practical solution, which has been adopted by some switch designers in the industry, is to design multiple memory modules to replace each single module. For example, the shared buffer in a switch will be implemented as a set of independent memory modules, which can all be read or written independently. Achieving good throughput is a challenge with such an architecture. We have developed a scheduling algorithm that one can use between the input ports and the memory modules, and also between the modules and the output ports. The operation of a switch with these multiple modules allows a scalable evolution of switch architectures to support very high-speed links such as fiber.

The fairness of scheduling disciplines has often been evaluated using one of two fairness measures: the absolute fairness bound, AFB, and the relative fairness bound, RFB. The RFB is the more popular measure since it is easier to obtain. We have established a tight bounded relationship between these two measures, and therefore, it is now easy to determine the latency of a scheduler directly from the RFB. Our results also indicate that in many real contexts, these two measures approach each other as the number of flows increases. This shows that the RFB, often considered an imprecise measure of fairness, is actually a perfectly adequate measure in real contexts.

We have also developed a novel scheduling strategy for use in systems where data blocks (for e.g., cells) can be forwarded independently, but where it is advantageous to forward related data blocks together (for e.g., cells that make up a packet). Such situations arise in Internet backbones with Internet Protocol, IP, implemented over asynchronous transfer mode, ATM, and also in wormhole networks used in multimedia servers where each packet is divided into chunks which are forwarded independently to achieve improved delay bounds. Our scheduling strategy combines a number of established principles to achieve both fairness and low delays in the resulting system.

We have significantly expanded upon our previous work on scalable fair scheduling algorithms, and developed scheduling strategies with much better latency characteristics. In the early part of this project, we developed the Elastic Round Robin, ERR, a fair scheduling strategy for best-effort traffic. At the time of its invention, ERR had superior fairness and delay characteristics in comparison to all other algorithms of equivalent efficiency. However, being a round robin strategy, it suffers from certain limitations of all round robin algorithms. One such limitation is the bursty nature of its transmissions which leads to large delay jitter values, thus rendering it less suitable for multimedia application traffic. In our latest work, we have developed a significant enhancement to our previous work, a new scheduling discipline that we call the Prioritized Elastic Round Robin, PERR. The ERR scheduler, in its normal operation, serves the active flows in a round robin order. In PERR, however, a prioritizing module is appended to the original ERR scheduler. The associated priority queues alter the transmission sequence of the packets in ERR by giving precedence to flows which are lagging in service in comparison to other flows. The total service received by a flow in a PERR round is the same as it would have received in the corresponding round if served by an ERR scheduler. However, in PERR, the service received by a flow is split into several parts over the course of a round. Our analysis shows that PERR results in a significant improvement in the latency and fairness characteristics in comparison to ERR as well as certain recently proposed algorithms such as Pre-order Deficit Round Robin. PERR is also easy to implement because of its low  $O(1)$  work complexity. PERR also achieves a significant reduction in the buffer requirements in comparison to Pre-order Deficit Round Robin.

We have also contributed to a growing body of literature on the practical aspects of scheduling algorithms. Fair scheduling strategies have often been designed with a focus toward minimizing the latency bound. However, many playback multimedia applications adapt their playback delay to currently observed jitter and delay characteristics in order to maximize the length of time over which the receiver of the multimedia stream obtains good quality. It is therefore important to consider not just the latency bound but the distribution of the latency achieved by any given scheduling algorithm. In the first work of its kind, we have determined the latency distribution of the Deficit Round Robin scheduler. We have shown that the latency distribution is primarily dependent upon the quanta used by other flows. In addition, we show that when the number of flows is large the distribution can be more easily obtained by a two-step approximation. Our work shows that the playback buffer used at the receiver can be significantly reduced if the latency distribution is taken into account, as compared to considering only the latency bound.

We have continued to make progress on practical scheduling algorithms to support QoS mechanisms and architectures. A popular measure of the fairness achieved by packet schedulers is the relative fairness bound, RFB, which captures the maximum possible difference between the normalized services received by any two flows. However, in the actions taken at packet transmission boundaries, currently known schedulers have generally used a goal other than explicitly reducing the RFB; for example, weighted fair queuing, WFQ, seeks to achieve the same order of packet transmissions as in an ideally fair fluid flow system. In our work, we have developed Greedy Fair Queueing, GrFQ, a novel scheduler that explicitly incorporates the goal of achieving a low RFB into the actions of the scheduler. We have proved that the RFB achieved by GrFQ is extremely close to that of the best among known fair schedulers. While worst-case fair weighted fair queuing, WF<sup>2</sup>Q, achieves the best possible bound on the service lag in comparison to a fair fluid flow scheduler, the GrFQ scheduler achieves a better bound on the normalized lag, and is therefore, more fair by this measure. We further argue that measures based on the RFB or on the lag are only bounds, and do not completely capture the actual fairness achieved at most instants of time. We borrow from the field of economics and use the Gini index to show that our scheduler achieves better fairness than any other known scheduler at virtually all instants of time with both synthetic traffic and real video traffic traces. The GrFQ scheduler is also computationally efficient since it avoids the emulation of a fluid flow system and has a per-packet work complexity of  $O(\log N)$  with respect to the number of flows.

The original motivation behind this work was the development of an overall goal-oriented strategy that may be employed to achieve any of several notions of fairness or any other target policy. The above represents our work in the evaluation of this strategy for max-min fairness as the target policy, which has yielded a computationally efficient scheduler with better fairness characteristics than any other known scheduler. Our work continues to employ this strategy to achieve admission control and scheduling policies beyond fairness.

We have continued to make progress on practical scheduling algorithms to support QoS mechanisms and architectures, and completed work specifically targeted for emerging QoS architectures in the Internet. Our work makes use of concepts in economics, such as social welfare, to create economically viable models of router behaviors. In addition, we have tried to employ pricing as an effective tool to control congestion and achieve QoS provisioning for multiple differentiated levels of service. We have significantly revised and refined our work on a novel, practical, flexible and computationally simple pricing strategy that can achieve QoS provisioning in Differentiated Services networks with multiple priority classes operating at close to peak efficiency, while also maintaining stable transmission rates from end-users. In contrast to previous work, in which dynamic pricing strategies are based on the state of congestion alone, our strategy adds a separate price component for the preferential service received by a packet. This permits an efficient market for network resources and services, with the price charged dependent upon both the cost of the resources and the dynamically changing demand for it. This automatically enforces efficient capacity management in the allocation of resources among the various service classes, also leading to a user-centric approach where a user is not charged a higher price unless preferential service is actually delivered.

## Insights derived and lessons learned

Our QoS research as part of the Pegasus project has led us to several engineering insights. However, the most important insight gained through the work conducted as part of this project is regarding the relevance of considering and incorporating economic incentives into the engineering design process. During the last several years, QoS issues in the Internet have attracted significant research interest and have resulted in at least a few important proposals for achieving QoS through innovative protocols, architectures and algorithms, including some that have resulted from the work conducted as part of this project. Unfortunately, we have noticed that a complex set of factors---the inertia of the marketplace driven by open, therefore somewhat frozen, standards, insufficient evidence of a strong customer base, the nominal adequacy of existing solutions and the highly competitive environment that emphasizes short-term performance goals for companies---have all together placed practical hurdles in the actual adoption of many of these proposals. Besides solutions based purely in engineering, a number of proposals have been advanced that seek to achieve QoS through economic means, such as through pricing. However, pricing strategies have not been studied within the context of engineering mechanisms that need to be in place to support the strategy through secure and scalable billing and accounting procedures. While solutions based in engineering alone are encountering difficulties in gaining widespread adoption due to a variety of reasons, solutions based on pricing alone have similarly gained no greater acceptance in the marketplace because they haven't considered the engineering hurdles involved. Good engineering design that has a true impact on the marketplace through actual adoption should consider built-in economic incentives in addition to offering engineering viability. As part of this project, we have proceeded in this direction and developed a practical, flexible and computationally simple and scalable pricing strategy that can achieve QoS provisioning in Differentiated Services networks with multiple priority classes at close to peak efficiency, while also maintaining stable transmission rates from end-users. Our work in this direction continues today and applies this insight in other aspects of networking besides QoS. For example, "route storms" and related routing inefficiencies in the Internet, we believe, can be substantially eliminated with a pricing framework among backbone providers that exploits economic incentives for carrying transit traffic.

In the following, we mention a few other insights gained through this project:

Much of the research on network considers the impact of self-similar characteristics of traffic on the performance of the network but not the impact of the network on the characteristics of the offered traffic. There exists, however, a complex interaction between the network traffic characteristics and the network at multiple layers of the protocol. Even at the simple link layer and the network layer, our research demonstrates that a packet switched network adds to the self-similarity of non-self-similar traffic and reduces the self-similarity of highly self-similar traffic. A closed-loop analysis of the interactions between the network and the traffic characteristics is essential to further our understanding of the implications of network design choices.

Fair queuing has been defined for a fair allocation of bandwidth on a link. As flows of traffic traverse through a network, they share with other flows a variety of network resources including links, buffers and router CPUs in their paths. Fair queuing is only fair when the link resource (bandwidth) is the only scarce resource. When buffers and/or the processing power

are scarcer, such as in optical networks, new principles need to be developed to achieve fairness and quality of service requirements. Another example is a situation in mobile ad hoc networks for military applications where both bandwidth and energy (for survivability) are scarce resources. When more than one resource is scarce, it may or may not be possible to trade off one resource for another, and the issue of fairness becomes significantly more complex. Fair queuing used in bandwidth allocation does not always lead to fairness, or reliable quality of service guarantees.

The current trend in QoS provisioning strategies is to avoid per-flow reservations to reduce the overhead of network management and to allow a more dynamic and efficient use of network resources. In the absence of reservations, the network devices have to implement policies to appropriately satisfy QoS requirements whenever the demand for a network resource exceeds the supply. These policies cannot merely degrade the performance of all users upon encountering congestion for resources, since it leads to a poor economic model. A good example can be found in telephone networks, where it is better to deny service to some users through blocking and serve admitted calls at a high quality, instead of admitting and serving all calls at a poorer quality. However, research efforts on policies for QoS provisioning have primarily focused on this model and tried to equalize the service received by all users (such as in fair queuing). An altogether new approach such as through a strategy to maximize social welfare (by maximizing the sum of the consumer surpluses) may lead to very different policies than those discussed in the research literature, and may actually be more efficient and more likely to be adopted when QoS provisioning for multimedia transmissions becomes significantly more important to Internet users.

### **c.3 Embedded Network Processing (Jonathan Smith, et. al.)**

#### **Background**

The research challenge was the mismatch between network performance and processor performance. While processor performance increases at an impressive rate (essentially doubling every 18 months), network performance as measured by the total capacity of a fiber-optic link (serial rate \* wavelengths) increases at a far greater rate, as each of the serial bit rate per wavelength, and wavelength count, have been improving at a rate more or less proportional to processor performance.

While network-embedded processing is attractive for a variety of reasons (e.g., latency, multiplexing, and resource management), applications other than security applications (such as the implementation of firewall functionality and intrusion detection) have been slow to emerge. Nonetheless, these security applications are important today and any design for future systems must take these user needs into account. We used, as a test case, the design of an intrusion detection system that is suitable for attachment to a wavelength-division multiplexed optical system.



## **ABIDE**

The Advanced Broadband Intrusion Detection Engine, ABIDE, is an architectural approach to the scalable intrusion detection system challenge. ABIDE structures a set of processors as a tree, in the style of the Columbia Dado and BBN Butterfly parallel computers. ABIDE's processors are heterogeneous: i.e., leaves are implemented by network processors, such as Intel's IXP1200, which perform initial selection and filtering of data from attached networks; and the inner nodes of the tree are implemented with general purpose processors. Many styles of specialized interconnection network are possible, but, as an initial selection, the gigabit Ethernet presents an excellent design point, with substantial throughput, low latency, and a large base of existing software, such as packet filtering libraries.

While ABIDE has not been evaluated in the context of an implementation, the major tradeoffs that should be investigated in such an implementation would be the throughput versus filtering rule and filter structure for data fusion versus topological structure of ABIDE. A final important architectural question is the software structure inside the network processors, a question that requires asymmetric multiprocessing (where both the processors and the tasks they execute are different, as is the case with the network processor leaf machines in ABIDE). We investigated this software structure question by construction of an operating system for network appliances, called Piglet.

## **PIGLET**

S. J. Muir's doctoral thesis discussed the design and implementation of Piglet, an operating system for network appliances. Piglet uses the basic architectural idea of using one logical pool of processors to do network I/O, and the other to perform general-purpose computation. A non-blocking synchronization method, polling, is used to control interactions among elements of the processor pools. Piglet was demonstrated on dual-processor Pentium machines, using Linux as the operating system for the general-purpose nodes and Piglet to carry out network I/O tasks. When Piglet was used for network appliance services such as web service (using the Rice "Flash" web server), it offered a considerable performance advantage over the same hardware using Linux in asymmetric multiprocessing style. What Piglet teaches us is that a software architecture for network appliances is facilitated with specialization. The Intel IXP1200 is composed of micro-engines for per-packet processing and a general-purpose processor for control and management functions. The Piglet system is extremely well suited to managing the micro-engines, and efficiently communicating between them and the general-purpose computing engine. This suggests that ABIDE is feasible and can, in fact, be programmed in such a manner as to provide intrusion detection at even the highest throughput rates.

### **Piglet: An Operating System for Network Appliances** **Steve J. Muir, University of Pennsylvania [34]**

#### Abstract:

Network appliances, such as web servers and routers, present operating systems with very different workloads than general-purpose applications. In particular, the predominant component of such workloads is data transfer operations, not computation. By designing a domain-specific

operating system, it is possible to take advantage of such properties to simplify the system structure and increase performance.

I have used this approach to design a new operating system, “Piglet”. Piglet is based on a new kernel-architecture: Active Kernel. An Active Kernel OS dedicates a subset of the CPUs in a multiprocessor system to running the kernel continuously; and uses polling as the fundamental communication mechanism for interacting with both applications and devices.

Piglet has been implemented as a hybrid Piglet/Linux system, using Piglet to provide the network subsystem but Linux for other OS services. This prototype has been used to evaluate Piglet using a variety of benchmarks, both micro- and application-level. These results show that the Active Kernel architecture performs significantly better, in both latency and throughput, than a conventional operating system (Linux).

## **Summary**

We have used network embedded processing as a design paradigm for building an advanced future network. We tested the idea by building a scalable intrusion detection system based on using demultiplexed WDM channels coupled to a parallel computer. Further, we have shown how system software for this architecture could be structured to deliver high performance.

## **c.4 IP over WDM (Mohamed Ali, et. al.)**

### **Background**

The evolving next generation Internet transport infrastructure is moving towards a model of high-speed routers interconnected by intelligent, reconfigurable optical core networks that will directly provide a global transport infrastructure for legacy and new IP services. In this model, clients (e.g., internet protocol/multiprotocol label switching routers, IP/MPLS) are attached to an optical core network, and connected to their peers over dynamically switched optical paths (lightpaths) spanning potentially multiple optical cross-connects, OXCs. The optical core network consists of multiple optical cross-connects interconnected by optical links in a general mesh topology. This transition is driven by the deployment of high-speed IP routers and ATM/MPLS switches. Specifically, these data-centric network elements, NEs, provide the essential bandwidth management functions, including multiplexing up to OC-12, OC-48, OC-192, GbE, and 10GbE rates. The use of interfaces with such rates renders intermediate SONET multiplexing NEs as potentially obsolete. This reduction in network overlays, and the associated elimination of SONET multiplexing and stand-alone NEs, is also accompanied by removing SONET-layer bandwidth management and replacing it with management at higher (IP, ATM) or lower wavelength division multiplexing, WDM, layers. In this scenario, the function of multiplexing traffic onto wavelengths may be passed on to the IP/MPLS routers.

A critical issue for realizing such intelligent optical networks is how to provide the desired features of rapid provisioning/restoration and automated capabilities between the optical layer and the client layers. It is widely accepted that the best way to achieve this is to adapt the IP topology self-discovery and routing capabilities to the optical network environment. Current research focuses on the use of distributed management schemes such as multi-protocol label

switching, MPLS, to provide the control plane necessary to ensure automated provisioning and maintaining connections and managing network resources. In this type of application the label is the wavelength of the incoming signal; hence, the term multi-protocol lambda switching, MP $\lambda$ S, is more commonly used.

Industry organizations like the OIF and IETF are rushing to extend MPLS-framework (Generalized-MPLS) to support not only devices that perform packet switching, but also those that perform switching in time, wavelength, and space. What remains a major open issue is how the control plane for the optical domain should interact with the IP (or other client) control planes. In other words, how IP/MPLS routers must interact with optical core networks to achieve end-to-end connectivity. There are three basic types of control plane options (three interconnection models) being considered to interface the IP domain to the optical domain: 1) the overlay model in which IP and optical domains have separate control planes and they do not share routing information; 2) the augmented model in which IP and optical domains have separate control planes, but they share routing information and each becomes directly involved in the others' routing scheme; and 3) the peer model in which there is one control plane that runs both the IP and optical domains.

This work has focused on developing and implementing comprehensive, unified constraint-based routing and signaling models and algorithms within the generalized MPLS framework, GMPLS, to provision a full range of bandwidth entities, e.g., packet flows, "sub-wavelength" circuits (low-rate traffic streams), and full wavelengths. Specifically, the implications of implementing the proposed algorithms for each of the three emerging interconnection models (the overlay, augmented, and peer models), as to which one may best fit a given set of traffic characteristics or a given application, is a key issue that has been addressed here.

## **Results**

We have been addressing the important issue of how both lightpaths (full wavelengths) and low-rate traffic streams (sub-lambda) can be provisioned automatically. Dynamic provisioning will enable the rapid provisioning of new connections. Activation times will drop to minutes. Lambda-based traffic segregation will enable the support of multiple qualities of service. Automated provisioning systems will gain direct access to WDM resources, and real-time dynamic lightpath provisioning will become feasible for accommodating vary traffic requirements and for recovery from losses of network assets.

Provisioning of connections requires algorithms for route (path) selection, and signaling mechanisms to request and establish connectivity within the network along a chosen route (path). Specifically, this work has addressed the implementation issues of both the path selection and signaling components of the traffic-engineering problem in such a network. In this work, provisioning of a connection request implies that a flow of data/an optical channel is successfully routed if both an active path (working) and another alternate link/node-disjoint path (backup) are set up at the same time.

During the first phase of this work, we addressed the implementation issues of the path selection component (first component) of the traffic-engineering problem in hybrid IP-centric DWDM-

based optical networks. Constraint-based optical path computation, is a special case of the routing and wavelength assignment, RWA, algorithm. We have presented and compared the performance of several different constraint-based routing and wavelength assignment, RWA, algorithms for dynamic provisioning of the optical channels. These RWA schemes are then used to compute end-to-end dedicated and shared backup paths to protect against single link/node failures. These schemes are based on fully distributed models, in which all optical nodes maintain a synchronized, and identical, topology and link state information base (traffic-engineering database, TED).

Specifically, unlike the conventional static RWA scheme used in most algorithms, which is often decoupled into the routing sub-problem and wavelength assignment sub-problem, the proposed algorithms integrate both the routing and wavelength assignment sub-problems into a single dynamic constraint-based routing problem. Thus, the emphasis here is on the adaptive routing problem, rather than focusing on the wavelength-assignment problem. It has been shown that the routing scheme has much more of an impact on the overall network performance than the wavelength-assignment scheme. Under this scenario, the overlay model is shown to be of interest in long-haul networks... where the traffic is characterized by large, homogeneous data flows and where the network is required to provide dynamic services at the full wavelength capacity (from OC-48 to OC-192 and on to OC-768 in the future).

In typical metro network scenarios, the traffic is dynamic and heterogeneous where the networks are required to provide dynamic services to the user at a rate that is much lower than the full wavelength capacity. These sub-rate streams are very common and can comprise smaller time domain multiplexing, TDM, channels (e.g., SONET/SDH 155 Mb/s OC-3 or 622 Mb/s OC-12), or a variety of storage area network, SAN, protocol interfaces (e.g., 200 Mb/s Escon, 100/1000 Mb/s Ethernet, 1.0 Gb/s Fiber Channel, etc), or even arbitrary packet flows. In addition, in networks of practical size, the number of source-destination traffic connections is still an order of magnitude higher than the number of available wavelengths. In order to achieve maximum efficiency, one would need to bundle these low-rate traffic streams efficiently onto wavelengths so the number of wavelengths that have to be processed at each router is minimized. The main objective is to reduce the number of IP ports needed on the routers (one per wavelength added/dropped at the router) as well as to reduce both the total switching capacity of the routers and the number of wavelengths required to achieve full connectivity.

To address this issue, we have extended the dynamic provisioning schemes developed so far (where the bandwidth of a connection request is assumed to be a full wavelength) to allow the provisioning of “sub-lambda” connection flow requests. In this case, our simulation results have demonstrated that low-rate traffic streams may best be dynamically provisioned at the IP/MPLS layer. To do so, one needs to design a survivable logical topology on top of the underlying physical connectivity that connects IP routers.

This is achieved as follows:

First, several low-rate data flows are statistically multiplexed (groomed) on to one wavelength at an IP/LSR router. Then, “conventional” dynamic lightpath provisioning schemes at the physical WDM layer, where the bandwidth of a connection request is assumed to be a full wavelength

capacity (overlay model), are extended to allow the provisioning of “sub-lambda” connection flow requests at the IP/MPLS layer. Specifically, we have developed a novel integrated protection scheme to dynamically allocate restorable-bandwidth guaranteed paths for designing IP/MPLS over WDM networks that can protect against single optical link/node failures. In this approach, the IP network routing, topology distribution, and signaling protocols are assumed to be independent of the corresponding protocols in the optical domain, but relevant routing information must be shared (augmented model). The direction for information flow is from the optical layer to the IP layer. This will allow the IP layer to make routing decisions and request connections (whenever needed) from the optical layer.

We have also developed several generalized multiprotocol label switching GMPLS-based routing algorithms that allow advanced quality of service, QoS, routing frameworks to be leveraged for sub-wavelength tributary routing algorithms, in addition to regular packet and lightpath routing and wavelength assignment schemes. Specifically, we have developed two different QoS-based routing algorithms to dynamically route low-rate traffic streams in future internet protocol/label switching router, IP-centric optical networks. One algorithm attempts to first route connection requests on the logical topology interconnecting the Internet Protocol/label switching router, IP/LSR, routers; and then, if blocked, on the underlying WDM physical topology by setting up a lightpath, if possible. The second algorithm attempts to first route connection requests on the physical topology, then, if blocked, on the logical topology. This approach increases the connectivity of the virtual topology. The performance of each algorithm is evaluated for three different QoS-based constraint-based selection schemes, namely, conventional shortest path selection scheme, least loaded selection scheme, and most loaded selection scheme.

In the second phase of our work, we worked on implementing solutions for addressing the second component of the traffic-engineering problem: the signaling component that can reserve resources and establish path state in the network nodes selected by the route calculation process. Specifically, we have developed four different distributed signaling protocols for fast automatic setup and tear-down of paths across the emerging interconnection models for IP-over optical networks. The first scheme is a flooding-based routing, FBR, algorithm with backward reservation while the second scheme is based on an adaptive routing algorithm called *Multi-Path Routing, MPR*, where  $k$  paths are probed simultaneously. Two path selection schemes are considered for the MPR approach: a first come first serve, FCFS, path selection scheme and a least-congested path, LCP, selection scheme. Our objective in developing these protocols is twofold: first, to avoid the implementation complexities associated with GMPLS-based constraint-based routing using label distribution protocol, CD-LDP, and resource reservation protocol – traffic engineering, RSVP-TE, signaling protocols; and second, to adapt the performance optimization algorithm to the requirements of different user applications by having the flexibility to vary the relative weight assigned to each of three performance metrics [call acceptance rate, CAR, call set-up time, CST, and routing distance, RD].

The main characteristic of the first two signaling schemes are: 1) the destination node makes, adaptively, both routing and wavelength selection decisions; 2) no global state information need be exchanged among network nodes for either scheme; 3) both schemes attempt to combine the benefits of both the conventional preferred neighbor (poor CST) and forward-based flooding

(poor CAR) approaches in a way to improve all the three performance metrics simultaneously; 4) both schemes are able to achieve a lower CST and RD compared to that of CR-LDP, RSVP-TE, and the preferred neighbor approach, due to their non-backtracking nature, since all paths are probed simultaneously; and 5) both schemes use backward-based reservation signaling to alleviate the excessive reservation of resources. This leads to a higher “call” acceptance rate (CAR).

The third and fourth protocols are GMPLS-based distributed control and management protocols. The third protocol is a global information-based link state approach that consists of both an integrated RWA algorithm and a signaling algorithm. Two triggering mechanisms for the LSA update procedures are considered; one is period-based and the other is a threshold-based update. The fourth protocol is a local-information-based fixed alternate link routing approach where the signaling protocol is closely integrated with the RWA protocols. No update messages are required in this approach.

### **Lessons learned**

- Reducing network overlays and eliminating SONET multiplexing and associated stand-alone NEs, by removing SONET-layer bandwidth management and replacing it with management at higher (IP, ATM) or lower (WDM) layers, is a critical component for realizing the Next-Generation Internet.
- A critical issue for realizing intelligent optical networks is how to provide the desired features of rapid provisioning/restoration and automated capabilities between the optical layer and the client layers. We are strongly convinced that the best way to achieve this is to adapt the IP topology self-discovery and routing capabilities to the optical network environment. This implies that there should be an IP-based control plane running the optical layer.
- Combining Optical Cross-connects, OXCs, with IP/MPLS routers is more economical than using a stand-alone router under most practical scenarios, reflecting volume of through and terminating traffic, statistical gains, and OXCs and router cost.
- The Overlay model is attractive in long-haul networks where the traffic is characterized by large, homogeneous data flows and where the network is required to provide dynamic services at the full wavelength capacity (from OC-48 to OC-192 and on to OC-768 in the future).
- The Augmented model is attractive in typical metro network scenarios where the traffic is dynamic and heterogeneous and where the networks are required to provide dynamic services to the user at a rate that is much lower than the full wavelength capacity. Under this scenario, low-rate traffic streams may best be dynamically provisioned at the IP/MPLS layer. To do so, one needs, to design a survivable logical topology on top of the underlying physical connectivity that connects IP routers. This is achieved as follows. First, several low-rate data flows are statistically multiplexed (groomed) onto one wavelength at the IP/LSR router. Then, conventional dynamic lightpath provisioning schemes at the physical WDM layer, where the bandwidth of a connection request is assumed to be a full wavelength capacity (overlay model), should be extended to allow the provisioning of “sub-lambda” connection flow requests at the IP/MPLS layer (Augmented model).

- Under this approach, the IP network routing, topology distribution, and signaling protocols are assumed to be independent of the corresponding protocols in the optical domain, but relevant routing information must be shared (augmented model). The direction for information flow is from the optical layer to the IP layer; this would allow the IP layer to make routing decisions and request connections (whenever needed) from the optical layer.
- Although this scheme is less complicated than in the peer, it still has the problem of sharing information across network boundaries, which will probably not be agreed to over public interfaces. However, we believe that there are a wide range of alternatives of what information to share and how these schemes would work.
- Most early work on traffic grooming was focused on SONET rings, where traffic is often static and known in advance. This is appropriate because today's backbone transport infrastructures are organized in rings. However, as networks evolve to become more IP-centric, grooming for IP traffic will become an important area for future work. In the IP environment, however, traffic is typically neither static nor known in advance. Furthermore, as network architectures transition from ring-based to mesh-based, grooming in mesh-based networks will become an important extension to current ring-based grooming algorithms.
- We envision that the full benefits of the optical Internet can be realized only when the capabilities offered by optical technology, available today primarily at the core of the network, spread toward the edges, extending the optical reach as close as possible to the end-user.
- Although most of the discussion within the IETF and OIF continues to focus on the distributed model, we point out to the potential role and advantages of having at least some centralized control capabilities to handle the complexity inherent in the optical layer, and argue that the most favorable long-term approach might include a combination of centralized and distributed control.
- It is important to point out that future networks will probably need a combination of these approaches - in other words, it's not an either-or situation. We see distributed techniques handling many basic lightpath set-up and tear-down, but centralized control handling complex restoration events, network policy enforcement, network optimization tasks and inter-domain routing that may be beyond the capabilities of distributed control software and protocols.
- There is, we believe, no single solution to the challenges associated with Internet scaling bottlenecks, but rather a combination of solutions is required to address the many different areas where the bottlenecks occur. Faster network equipment, larger pipes, smarter software, and localized content all are important pieces in a larger puzzle.

## **c.5 Network Traffic Modeling (Stewart Personick, et. al.)**

### **Background**

Recent studies of high-quality, high-resolution network traffic measurements have revealed that modern packet traffic appears to be both self-similar (or long-range dependent) and non-Gaussian distributed. In order to realize desirable properties in communication networks related to quality-of-service, QoS, it is crucial, in traffic engineering, to develop efficient traffic models that are capable of yielding acceptably accurate performance predictions in a reasonable amount

of computational time. We have shown evidence that although heterogeneous network traffic has complicated short-range and long-range temporal dependencies, for certain models of self-similar network traffic, the corresponding wavelet coefficients (using wavelets as basis vectors to simulate the traffic) are no longer long-range dependent. Indeed, the multi-scale property of wavelets may make wavelet representations a natural and powerful analysis/synthesis tool for network traffic modeling. Therefore, although still remaining as a challenging problem, modeling dynamic network traffic in the wavelet domain may be significantly easier than in the time domain. We have initiated an activity to develop stochastic models for self-similar traffic based on wavelet analysis/synthesis techniques.

## **Progress**

We studied the characteristics of several, alternative wavelet models of packet traffic. The models we tried to use all have revealed significant shortcomings, including shortcomings in their ability to capture the second order (spectral or correlation) properties of measured traffic. We considered alternative wavelet models and we tried to define measures of the suitability of various traffic models in real applications. Suitability is defined here as the ability of the model to predict the performance of the network with sufficient accuracy under various assumptions regarding the importance of various network behavioral characteristics.

We have produced some strategies for designing networks that attempt to anticipate and automatically accommodate (e.g., by provisioning buffer capacity and reconfiguring the network in near real time) statistical traffic fluctuations within certain time scales (e.g., <10 minutes), while employing ad-hoc techniques (e.g., intervention by traffic managers to choke off the sources of emerging overloads) to deal with fluctuations on longer time scales. These methods are based on the premise (to be verified) that long term fluctuations, that cannot be efficiently managed by provisioning and traffic rerouting, will provide enough advance warning to allow ad-hoc approaches to be effectively employed.

## **c.6 Wavelength Agility and Optical Networks (Bahram Nabet, Janet Jackel, et. al.).**

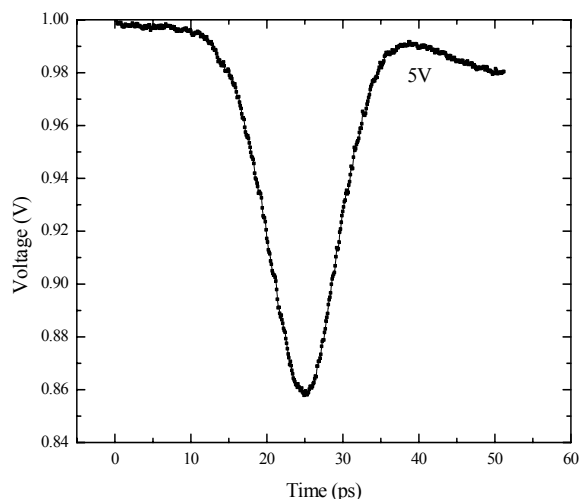
The objective of this research is to determine the viability of designing and implementing networks that employ a layered hierarchy of: packet switching, flow switching, add-drop multiplexing, wavelength routing, routing of groups of wavelengths, and switching of entire optical fibers, with the objective of reducing the cost and complexity of network traffic and quality-of-service management.

**Subsequent to the tragic events of 9/11, it became apparent that reduced complexity of network management would be a key factor in the ability to restore critical services in heavily damaged networks. Thus recoverability and robustness become major, driving forces in the process of network design and optimization.**



## Wavelength-selective optical detector (B. Nabet, et. al)

We previously reported fabrication of a novel photodetector that offers wavelength selectivity needed in optical networks, particularly in gigabit Ethernet. This tuned detector, which is based on vertical cavity surface emitting laser, VCSEL, structure, rejects out-of-band signals, and noise, and is particularly useful in broadcast-and-select switches. Figure 4 shows the device response time. In this reporting period we completed high-speed time domain measurements of the device. The source was a femto-second Ti:Sapphire laser and the data was collected with a sampling scope of 50 GHz bandwidth. Time response is shown in figure 4, indicating fall and rise times of about 7 picoseconds and a full width half maximum, FWHM, of about 8 picoseconds. The data indicates that our device is limited by the measurement system (probe tips, bias tee, scope) and has a potential bandwidth that may exceed 40 GHz. We have designed a new mask set that will allow us to incorporate an on-chip transmission line and perform higher resolution electro-optic sampling measurements. Telcordia Applied Research has expressed interest in characterizing and using these devices for their applications.



**Figure 4: Time response to femtosecond laser with wavelength of 850 nm**

## Wavelength agility in optical network elements ( Janet Jackel, et. al.)

### Background

Networks with multiple levels of granularity have been proposed by a number of groups<sup>1</sup> recently as a means of coping with the large numbers of wavelengths projected to be needed to realize an optical network with ultra-high capacity. In publications (Ref. 1, and also IPO Working Group Internet Draft on Optical Multi-Granularity, multi-granularity has been

<sup>1</sup> For example, Noirie, et al, "Impact of intermediate traffic grouping on the dimensioning of multi-granularity optical networks" OFC2001 paper TuG3, Noirie, et. Al., "Multigranularity Optical Cross-Connect," ECOC2000 paper 9.2.4

described as a means of reducing the size of optical switching matrices, thus reducing network cost.

<http://search.ietf.org/internet-drafts/draft-dotaro-ipo-multigranularity-00.txt>

Published discussions of multi-granularity, however, have not looked in depth into the physical implementation viability of the switches that are proposed. There appears to be an assumption that separating a set of wavelengths into bands, and combining bands of wavelengths into an aggregate, is straightforward and will introduce no signal impairments.

There are also unspoken implications:

1. That the number of ports in a switch for  $N$  fibers carrying  $M$  wavelengths is  $(NxM) \times (NxM)$  if wavelengths are switched or that for  $N$  fibers carrying  $B$  bands is  $(NxB) \times (NxB)$ .

In fact the first is true only if wavelength conversion is provided for all inputs on any wavelength. Without wavelength conversion  $M$  independent  $N \times N$  switches are needed. For band switches/crossconnects, there is no wavelength conversion<sup>2</sup> and the switch/crossconnect needed for  $N$  fibers with  $B$  bands would consist of  $B$  independent  $N \times N$  switches. In either case, the larger number of smaller switches could be realized using the larger switch, if such is available and economically preferable, but at this time there is an advantage to being able to use smaller, more readily available switches.

2. If only a specific and limited set of wavelengths can be switched at a node, the value of multi-granularity can be defined. If only a specific and limited set of bands can be broken into wavelengths, multi-granularity will be economically favored (assuming that performance is not affected); but if we demand that any band be able to be broken into wavelengths (more flexibility), then even if only a single band at a time is able to be broken down, filters will need to be provided for all, eliminating any economic advantage.

There may, however, be performance advantages to wave banding, even where there are no cost advantages. If a band is not broken down into individual wavelengths except when needed, those wavelengths do not incur the same performance costs (for example passband narrowing) in the nodes where they pass through intact.

3. It is assumed in publications that a band approach will introduce no impairments greater than those already in a network with each wavelength switched at each node.

This is not guaranteed. It depends strongly on the choice of wavelength multiplexing/demultiplexing technologies.

## Results

---

<sup>2</sup> Band wavelength conversion is not impossible, but it is complicated, involving nonlinear optical interactions, and has not been demonstrated with useful efficiency. Furthermore, band conversion lacks flexibility in wavelength assignment, unless multiple stages are used. Multistage band conversion is more complicated and likely much more costly than demultiplexing the band and converting each wavelength separately.

Based on the points above, we proceeded to demonstrate a band add/drop and a band cross-connect, using components available commercially at this time. Bands can be implemented either by combining an existing set of wavelengths into coarser granularity or by subdividing an existing set of wavelength channels to provide finer granularity. We pursued the latter approach for two main reasons:

1. Multi-granularity has its greatest advantage when the number of channels becomes large. This will occur only for finer channel spacings than are currently implemented in most available WDM networks. We are now working with a legacy tested (MONET NJ testbed), which has 8 wavelengths on 200 GHz centers. Grouping these wavelengths into bands can have an advantage to the extent that it reduces the number of switch fabrics needed, but the advantage is small compared with that gained in a network with larger number of wavelengths.
2. Networks with a small number of widely spaced wavelengths use the available bandwidth inefficiently unless each wavelength channel carries an extremely high bit rate. Subdividing the bandwidth via multi-granularity allows more efficient use of the bandwidth.

We therefore concentrated on defining an architecture for a band add/drop and a band cross-connect, both of which assume no wavelength conversion, and which can therefore use an optical cross-connect of small size for each wavelength or band.

Candidate architectures:

Several candidate architectures were identified. Each of the choices shown has certain advantages and disadvantages:

### **Broadband fiber Bragg gratings**

A wave band add/drop based on broadband fiber Bragg gratings (Figure 6a) is simple to construct. However, we have found several performance problems associated with the gratings.

### **Bragg fiber grating filters to define narrow wavelength channels within the bands**

We ordered a set of these with 25 GHz channel spacing. The Bragg gratings are used in conjunction with optical circulators, which were also ordered. Because of the cost of these components, we utilized a minimal set to test the concept and to characterize optical impairments. (We used funds from another source to purchase an additional set of narrow Bragg filters and circulators, and thus were able to build a more realistic network.) We also purchased custom built, spectrally wide Bragg gratings to define bands.

From the process of working with the supplier of our wide band Bragg gratings, we find that:

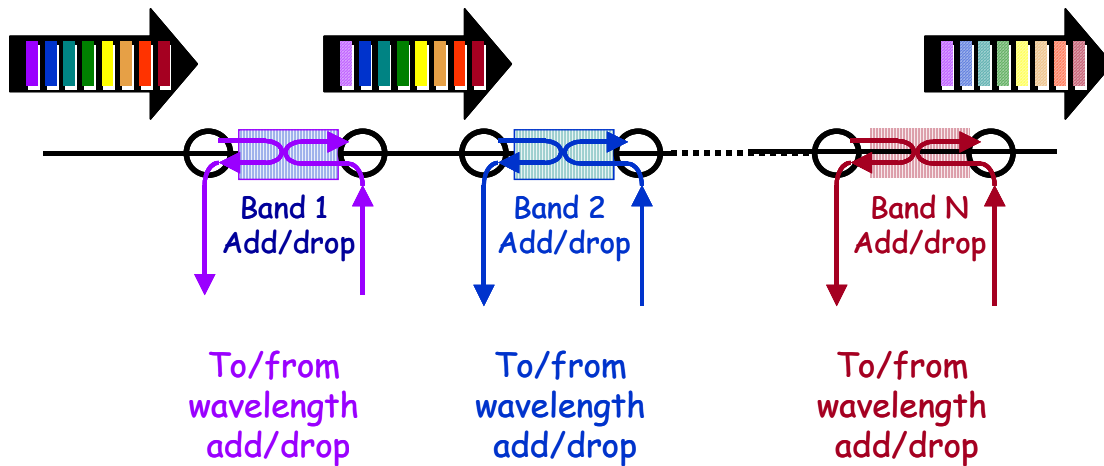
- a) It is difficult to produce a fiber Bragg grating with a very wide passband and an abrupt transition from transmission to reflection.
- b) There are tradeoffs between the width of the reflection band, the abruptness of transition, and unwanted reflections in the transmission region.

- c) For these reasons, the gratings are likely to be more expensive than we (or the supplier) had anticipated.
- d) Even with non-standard fiber, it will be hard to produce gratings with passbands greater than about 150 GHz.

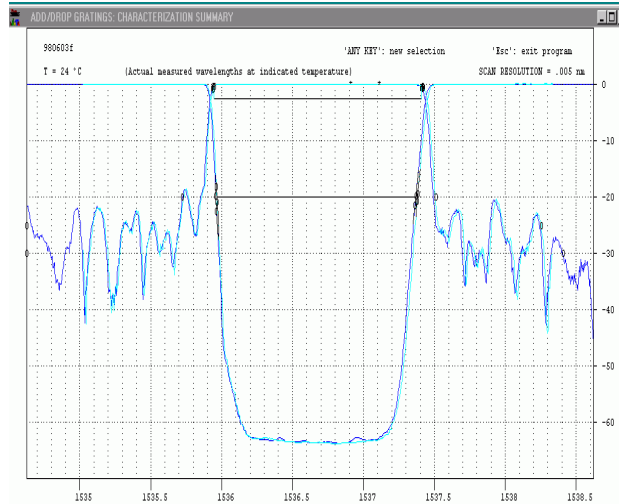
On the other hand, the narrow band (25 GHz spacing) fiber Bragg gratings are probably adequate for wavelength selection after a band has been separated out. Those we have purchased have approximately 10 GHz of passband.

We have also considered the use of interference filters for different wavelength granularities to be used in the same architecture. We now have filters appropriate for bands with 200 GHz spacing. We intend to use these along with the interleavers to define bands. However, these have narrower than desired spectral passbands, and we are also looking at the use of either wider interference filters (no longer readily available since commercial emphasis has moved to narrower wavelength spacing) or custom designed wide band fiber Bragg filters (see below).

Based on measurements of passbands, we do not believe that interference filters will meet the needs of either waveband add/drops or narrow wavelength separation. Wide band interference filters that might be suitable for waveband add/drop functions do not have sufficiently sharp transitions).



**Figure 5a: A waveband add/drop that is constructed from wide passband Bragg gratings**



**Figure 5b: Measured transmission and reflection for a fiber Bragg grating with a performance appropriate for a waveband add/drop**

#### Assessment of the architecture of Figure 1:

This add/drop architecture can be constructed from components now available, but it has a number of performance problems:

- Even with ideal filter performance, impairments arise because wavelength selection is passive and therefore all wavebands must be dropped and added even when the entire band is to pass through the network element.
- As the number of wavebands increases, the loss through this type of network element will increase, since all light must pass through a larger number of waveband add/drops.
- Waveband width is fixed; it is not possible to reconfigure with waveband itself.

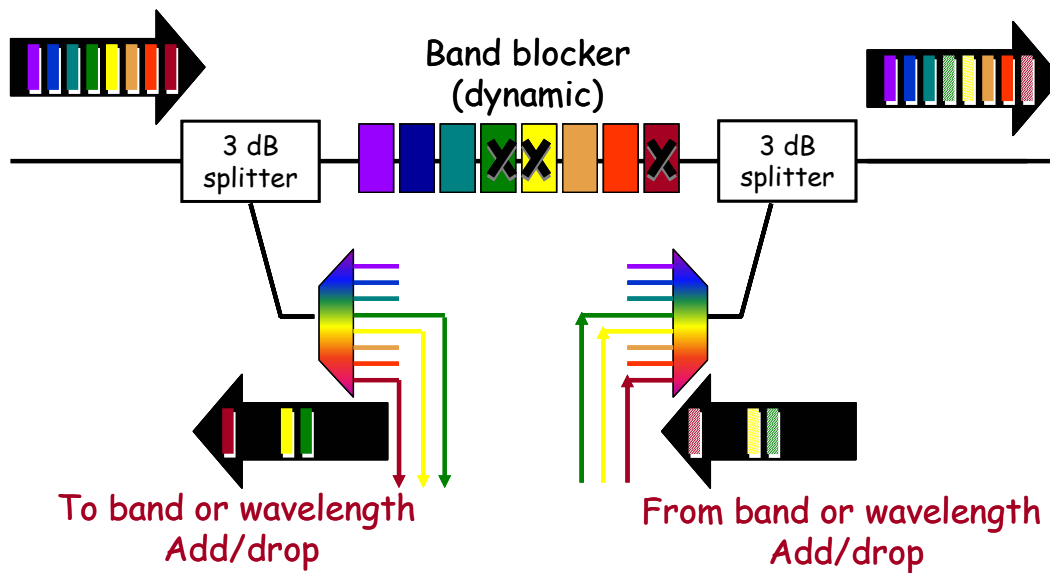
For these reasons, we do not believe that this architecture is suitable for any but the simplest multi-granularity network elements.

#### Dynamic band blocker

An architecture based on a dynamic band blocker (Figure 6) seems perverse at first, but has a number of advantages. In this add/drop, the incoming optical power is split into two paths. A spectral band blocker is placed in the through path; the drop path goes to a band demultiplexer, which can be built in any of a number of ways.

Clearly this guarantees at least 6 dB of loss in the add/drop for the through channels, plus the loss in the band blocker. In our lab the band blocker is a liquid crystal based switch<sup>3</sup> with broad ( $\geq 1.2$  nm) and flat passbands, and spectral efficiency greater than 80%.

<sup>3</sup> J. S. Patel and Y. Silberberg, "Liquid crystal and grating based multiple-wavelength cross-connect switch", *Photonics Technology Letters*, vol. 7, No. 5, p. 514-516, 1995.



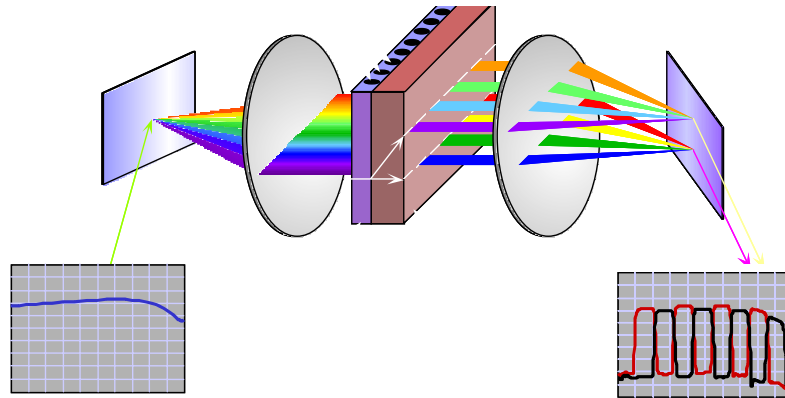
**Figure 6: Waveband add/drop based on a dynamic band blocker**

Figure 7 is a “cartoon” depiction of the switch showing its basic functions: gratings disperse the incoming spectrum, an array of liquid-crystal cells rotate polarizations of selected band, and polarization optics converts polarization rotation into displacement. The same structure can be used as either a dynamic band blocker or as a band add/drop, and can also function as a waveband interleaver.

The band blocker we use is based on an old Telcordia design, originally for the DARPA-funded, multi wavelength optical networking, MONET project, and is therefore designed for 8 wavelengths with 200 GHz spacing. A similar component is now made by Corning and sold as a “Pure Path” switch and is incorporated in some Marconi optical network elements.

This architecture has several advantages:

Only the wave bands that are to be dropped will experience spectral narrowing. The “drop and continue” function is trivial to implement, since all wavelengths pass into the “drop” port. The beam blocker can also act as a per-band power leveler, if monitoring is provided. Because wavebands are processed in parallel rather than in series, loss in the beam blocker is not strongly dependent on the number of bands.

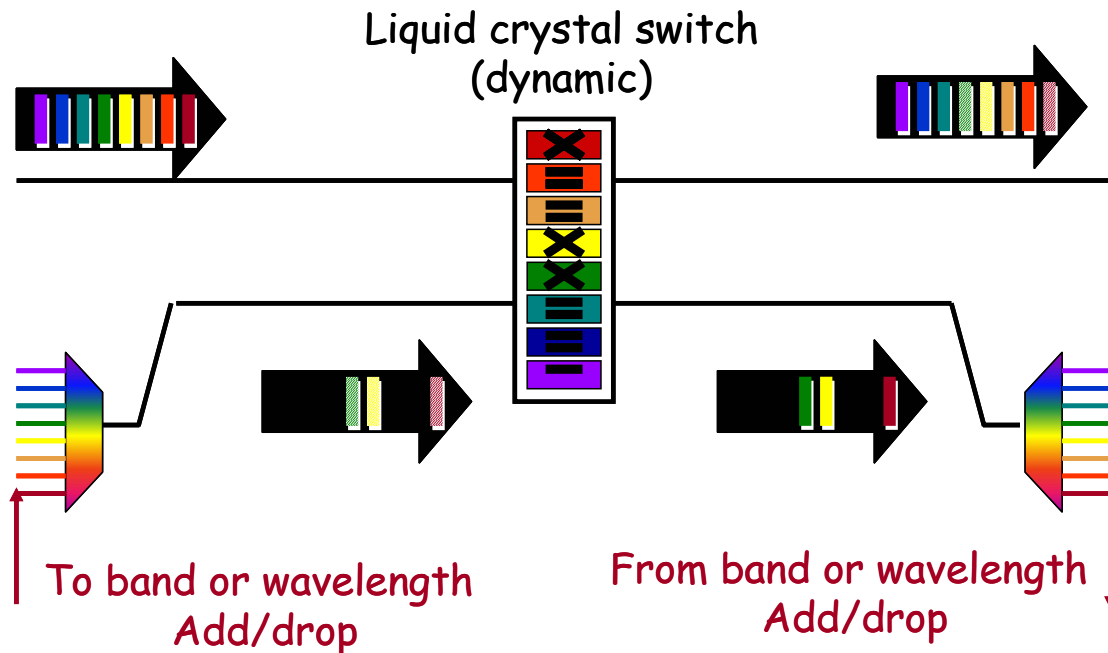


**Figure 7: Schematic picture of liquid crystal switch used as dynamic beam blocker or band add/drop. Input and output spectra are shown.**

The liquid crystal switch has intrinsic losses due to the complicated optical path. Another way to realize the same function is to replace the liquid crystal cells with a one-dimensional array of MEMS tiltable mirrors. This eliminates the multiple polarization selective optical components needed in the liquid crystal embodiment of the blocker, and should therefore reduce loss.

### Dynamic band selector

A network element architecture based on dynamic band selector uses a the liquid crystal switch or a similar component as a band add/drop. The switch drops (and adds) selected bands and all others pass through unaffected. The intrinsic through loss of this architecture is six dB less that that of the previous one, since the splitter and combiner are not used. This architecture has the same advantages at the previous, except that the neither the drop and continue function nor power leveling are possible without additional components.



**Figure 8: Band add/drop based on a the liquid crystal switch**

### Other technologies

#### Wavelength conversion:

At the wavelength granularity level, we have completed a comparative study of wavelength conversion techniques (see attached table) showing that our design is competitive in performance while being very cost effective. Components including agile tunable lasers and modulators have been ordered, with delivery date of mid February 2002. Test equipment have been assembled to measure performance characteristics, particularly bit rate transparency limits. During this phase we procured the most viable commercially available component that monolithically integrated the EAM (electro absorption modulator).



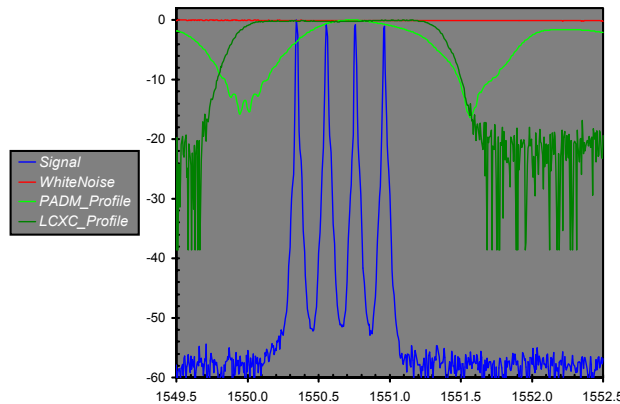
	OEO	FWM - Passive	FWM - Active	Cross-Gain/Phase Modulation - SOA	Diff Freq in Semi-conductor WG	EAM	(Saturable Absorber)
<b>Transparency</b>	Limited	Fully	Fully	Limited	Fully	Limited	Limited
<b>Conversion efficiency</b>	10-20dB	1.2% (-20dB)	-7dB	8dB GC-SOA - 2dB CP-SOA	-4dB theory, -17dB meas.	-20dB	
<b>Conversion Bandwidth</b>	>100nm	<20nm	~40nm		>90nm	>100nm	>100nm
<b>Signal Quality</b>							
Noise Figure	~8dB		SNR<20 [Yoo]	~8dB GC	same as input	? ~20dB	
Input Dynamic Range	~20dB, f(Rx)	>10dB	>10dB [Yoo]	3dB range for CP-SOA			
Extinction Ratio	10-20dB, f(Rx)	same as input	same as input	<8dB GC >12 CP	same as input	~12dB	>20dB
Dispersion problem		no	no	yes GC-SOA No CP-SOA	no	no	yes DM-DBR ?others
Distortion							
Phase preserved?	No	Yes	Yes		Yes	No	No
<b>Power Consumption</b>	2W						
<b>Bit rate limit/RF BW limit</b>	~10GBps						
<b>Tuning Speed</b>		Depends on tunable filter			Pump is short wavelength (~1/2)		
<b>Polarization Dependence</b>	No	Yes	Yes	design active region	No		
<b>Advantages</b>	* commercially available		*Transparent,				
<b>Disadvantages</b>	*limited transparency		*SNR degradation *Narrow BW				

Table 1: Summary of Wavelength Conversion Alternatives

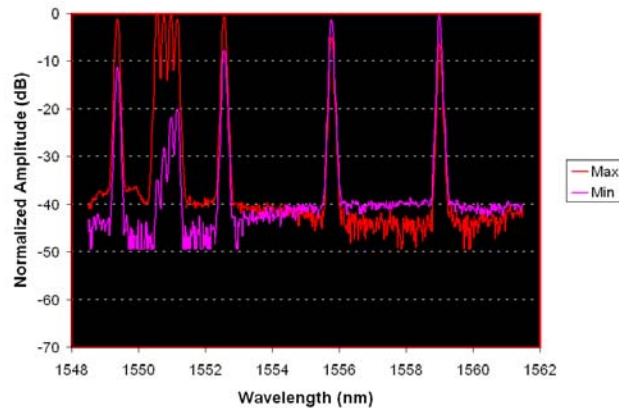
Some of the issues we have investigated include, (posed as questions/answers),:

1. How can individual wavelengths within a band be guaranteed to maintain acceptable optical power levels and optical signal to noise ratios when the entire band, rather than individual wavelengths, is being controlled?

We have looked at how passband shape affects this. Figure 9 shows the spectrum of a four-wavelength source with 25 GHz spacing compared with the passband of one band of the liquid crystal switch and that of a wide interference filter. Clearly the flatter passband will have less of an effect on the relative powers in the four wavelengths.



**Figure 9: Comparing the passbands of two types of band selectors with the spectrum of a four- wavelength source used to fill the band.**



**Figure 10: Power in each of two orthogonal polarizations for wavelengths within a band and in different bands, after 76 km of fiber in a laboratory, where all wavelengths entered the fiber in the same polarization.**

2. What effect will polarization evolution in the network have on the above? (Changing polarization, combined with polarization dependent loss or gain, can change relative powers within the band.)

We have made measurements of polarization evolution within and between bands, in the lab and over fiber only, i.e., with no network elements besides the first, which multiplexes the wavelengths onto the fiber. The consequences of polarization evolution within a band depend on polarization dependence of other components. Organizing the spectrum into bands and controlling optical power only at the granularity level of the band will enforce more stringent demands on the polarization independence of components.

3. How should bands be organized (contiguous wavelengths, interleaved wavelengths) to take advantage of the capabilities of the components? How will optical impairments differ for different choices?

Example: bands generated by interleavers are easier to break into individual wavelengths; however, widely spaced wavelengths would require flatter EDFA gain spectra. In addition, polarization evolution measurements (above) argue strongly against interleaved wavelengths. For the above reasons we believe contiguous bands are best.

4. How can monitoring be implemented in such a network?

Monitoring is understood to be a problem in any transparent optical network. It will be more difficult in one with multiple levels of granularity. Can we dispense with individual channel monitoring except at the edges if the network is well engineered? Monitoring remains an open question.

5. How does wavebanding affect network robustness?

In some cases, failure of a single component can affect larger amounts of bandwidth. On the other hand, multi-granularity affords opportunities for greater redundancy and flexibility in the network. Because fewer components are involved in reconfiguration needed for restoration after a failure, multi-granularity can make recovery from failures easier. This remains an open question. It is critical to investigate real scenarios for response to failures and how different switch designs affect vulnerability.

## **Lessons learned**

1. From an ARCHITECTURAL perspective, wavebanding would appear to be a natural way to dramatically increase the bandwidth capacity of an EXISTING transparent optical network by simply packing a larger number of wavelengths into a given optical channel band. However, when it comes down to implementing it in practice, the transparency of a given optical channel may not be as “transparent” as one might have expected (due to filtering & polarization effects, for example), thereby reducing the potential number of additional wavelengths that can actually be deployed. Therefore, during design of future optically transparent network elements, one should assess the impacts and tradeoffs of supporting wavelength banding, even if the network element is not intended to be used in that manner.

initially. Certainly we have established the need for different requirements on the network in order to accommodate multigranularity. Polarization effects are an example. Amounts of polarization dependent loss PDL that would be acceptable in standard WDM networks can be unacceptable in a network with wavelength banding.

2. Different switch architectures have very different consequences for multi-granularity.

Performance: Architectures which drop bands only when those bands either to be dropped at that node or are to have further processing (extraction of one or more wavelength) impose less band narrowing on the through traffic, and thus create less performance impairment. This has an effect on scalability of such networks, since the through traffic can pass through a larger number of nodes before performance becomes unacceptable.

Scalability: Architectures in which bands are processed in parallel (the second two we have shown, or any relying on arrayed waveguide gratings to separate bands) can more easily grow to a larger number of bands.

Cost: The primary cost advantage multi-granularity comes from reducing number of components. Only architectures that actually do so will reduce costs; if any wavelength can be dropped at any node, the components needed to do so will need to be present, even if they are seldom used. Consequently, the cost advantages of multi-granularity will require wavelength allocation that avoids this, and there will be some loss of flexibility.

3. Network robustness: While the failure of a single switch can affect more bandwidth in a network with multi-granularity, we believe that wavebanding will make it simpler to recover from many failures. A true understanding of the impact of multi-granularity on robustness requires further study.
4. Not all the components needed to build multi-granularity are available at this time. If we intend to build such networks we will need to stimulate their development.

## **D. Bio-informatics and Bio-complexity Applications**

### **d.1 Distributed Neuroanatomical Database (J. “Yoni” Nissanov, et. al. and Susan Davidson, et. al.)**

The goal of this research is to use high-speed networking technology to implement a distributed genomic-neuroanatomical database. The system will combine databases and software developed at the Center for Bioinformatics at the University of Pennsylvania (PCBI), including databases of genetic and physical maps, genomic sequences, transcribed sequences and gene expression data, all linked to external biology databases and internal project data, with mouse brain atlas data and visualization packages developed at the Computer Vision Center for Vertebrate Brain Mapping (CVCVBM) at Drexel University. The biological/medical value of the activity lies in the ability to correlate specific brain structures with molecular and physiological processes.

MacOStat, the atlas visualization software, was further developed during this past fiscal quarter. Image and volume-of-interest, VOI, can now be reliably transferred across the network. A problem that previously contributed to periodic fatal errors has been corrected. The current version has been enhanced to support multiple simultaneous session within a single window that supports co-navigation through multiple volumes. This will allow users to view multiple atlases, each of a different modality. Even of greater value, this functionality will allow, with the future expansion of Gigabit networks, simultaneous access to atlas and multiple volumes from the Mouse Brain Library, a repository of neuroanatomical mouse data being built as part of the Informatics Center for Mouse Neurogenetics (a National Institute of Health NIH supported program project on which Dr. Nissanov participates).

With the current MacOStat version, VOIs resident on the client side can be overlaid on all remote and local volumes. During the next quarter this function will be enhanced to support projection of remotely stored VOIs onto all displayed atlases. As planned, aids for performance monitoring have now been implemented. These report both network transfer speed as well as 3D-reslice time. The network, which as of the last progress report was operating at far below expectations, is now running at 120Mb/sec.

With the UPenn and Drexel link working correctly and the software reliably running over the network, we can now visualize 3D volumes in a client-server mode. A user sitting at UPenn is able to manipulate a data volume resident on the Drexel server. In the future, interaction with GUS, a system of combined databases, and software will rely on the K2 data integration software; we have therefore installed the K2 data integration software on the UPenn MAC during this quarter. Since this is the first time K2 has been installed on a MAC machine (all previous ports were to Linux or Unix machines), some troubleshooting was necessary. The problem that we are now addressing is the lack of support for simultaneous gigabit and Ethernet connection under MacOS 9.X. The most attractive solution is upgrading to MacOS X that does support dual networks. This change will require some modification in MacOStat.

We have demonstrated remote access to large, distributed bio-informatics data bases (including remote microscopic access to archived slides of mouse-brain tissue stored on a remotely controllable carousel slide server) using very-high-speed networking (> 100 Mbps), and we are

now planning to produce a CD-ROM that captures this very impressive presentation/demonstration in order to facilitate promulgation of our results. These demonstrations illustrate (in compelling ways) how research on the underlying genetic origins of brain diseases (and potential cures) can be facilitated and accelerated using very high-speed wide area networking.

From the perspective of the end-user, visualization of fused spatial and genomic data will make use of MacOStat, the atlas navigation software. This application was further developed during this past fiscal quarter. The MacOStat client has been modified to operate in a passive mode; at startup time, the client will automatically login to the atlas server, and then wait for data to be pushed from the server, in effect reversing the usual client-server relationship. This will allow the K2 data integration software to instruct the MacOStat server to send image data to the MacOStat client asynchronously, allowing the results of queries to the GUS database to be displayed graphically on the MacOStat client. In addition, the MacOStat client was modified to operate in the MacOS X environment, which should simplify the support of dual network connectivity.

In addition, the K2 data integration software was successfully ported from a Unix to Mac environment, as well as drivers to the GUS database. Drivers to the MacOStat server have also been written, and are being tested. A web interface has been designed and implemented which displays a taxonomic tree of mouse brain terms, and automatically fires off queries to GUS about gene expression within a selected portion of the taxonomic tree. The interface also allows the user to ask MacOStat to display the color-coded selected regions. This provides a proof of concept for the integration of sequence databases as well as specialized data sources and visualization software in the Mac environment using high-speed as well as normal Internet networks.

In 4Q2002 (calendar year) we successfully integrated the Neurocartographer software with K2 software and hypertext preprocessor, php, scripts to access the GUS databases, and display the results in graphical format via Neurocartographer. The integrated software allows the user to browse the GUS database, choose a sequence, and have areas where the sequence is known to be expressed viewed as highlighted on the Neurocartographer client display. The highlighted areas are colored based on expression data obtained from GUS. The user can then use the Neurocartographer client to browse the full atlas. In addition, the Neurocartographer server has been made more robust, resulting in improved up time. This work was demonstrated at the Next Generation Internet Principal Investigator meeting on January 6, 2002 in Tyson's Corner VA

## **d.2 Cellular Observatory (J. Yasha Kresh, Banu Onaral, et. al)**

Unraveling the complex dynamics of cellular network behavior is the principal focus of this study. Gaining insight into cellular organization requires the establishment of novel imaging techniques that will enable the study of functional cellular interactivity that has been not possible to date. Existing methods monitor singular cellular / sub-cellular events and/or produce only a static 'snap-shot' of a specific moment of a dynamic process of interest. Conventional "fixed-cell" assays are inherently deficient in elaborating the dynamic links between the inter- and intracellular processes that define cellular homeostasis. In contrast, a simultaneous tracking of a

set (e.g., three or more) of cell specific functions in each of the ‘units’ of the cellular assembly will help unravel some of the critical links that are involved in the dynamic re/organization of cellular network behavior.

The specific methodological approach to be implemented for identifying the cellular organization will involve binding of fluorescent markers to specific organelles within the living cell and recording real-time images of cellular network activity, spanning many hours of observation. The use of multiple fluorescent markers will enable to interrelate cell function (e.g., cytoskeletal meshwork, and second messengers) to the functional inter-connectivity of the cellular network, as well to its viability. The acquired images will form the basis for the time-lapse mapping of processes and emergence of intercellular communication patterns. The sequential frames (pattern formation) will be used as a baseline to study the role of spatio-temporal organization of the formed cellular network in adaptation to the imposed environmental changes (e.g. stress). The specific aims of the “Cellular Observatory” are as follows:

- Demonstrate a high speed multi-mode cellular imaging and digital microscopy collaborative
- Study live-cell spatio-temporal organization in real-time
- Study the role of intercellular communication and its effects on viability and “cell culture” survival
- Develop computational models to study cellular network viability and adaptability to imposed stress conditions.

## **Part I - Telemic Software Development.**

A module for interfacing and controlling remotely the Firewire (IEEE-1394) camera was completed. Importantly, the telemicroscopy ‘Server’ can capture images from the high-frame rate (200-400 Mbps) video color camera and write it to disk in jpeg format. The images can be captured to disk in a streaming manner (e.g. every 100 msec ). There is an added feature in place, such that only a change in image content (pixel-by-pixel comparison) necessitates the ‘Client’ to update the viewed image. Functionality includes <Filter Wheel Control> and <Shutter on/off Control> thereby enabling a remote user to change the emission filters and /or excitation filters at will. One can also close the shutters to prevent light bleaching (e.g. fluorescent dye) of cell culture under prolonged observation.

The X-Y stage movement control module is fully operational with the following features:

Incremental (0.2 micron / step) movements in the X-Y directions. i.e. NW, SW, SE, NE, E, W

Ability to move back to the set origin from any given position

Stage Calibration: This moves the stage to its limits and calibrates the stage.

Ability to move the stage to any specified X-Y co-ordinates, within the available field of excursion. This functionality is graphically emulated on the screen that maps the area available for stage movement and then provides cursors to mark the x-y co-ordinates. All one has to do is to drag the cursors and the stage moves correspondingly to the specified co-ordinates.

All the stage movements can be undone by clicking the UNDO button on the client window. The speed of the stepper motors is now controllable by the 'Client' -the stage can be directed to move at maximum or minimum speeds.

The user can snap an image i.e. capture a scaled version of the image from the camera for future reference. The focus control (Z-axis) module has been implemented. The client interface and functionality was changed from increment-decrement buttons to a 'slider' that changes focus in a more intuitive manner. This is like actually rotating the knob. The <EDIT> feature has been incorporated into the Telemic. By invoking it, one can capture an image from the camera and then edit it. Functions currently available are <Grayscale>, <Invert>, <Blur> and <Contrast>. This module can be expanded to include various nonlinear filtering algorithms and users may desire routine image processing functions as.

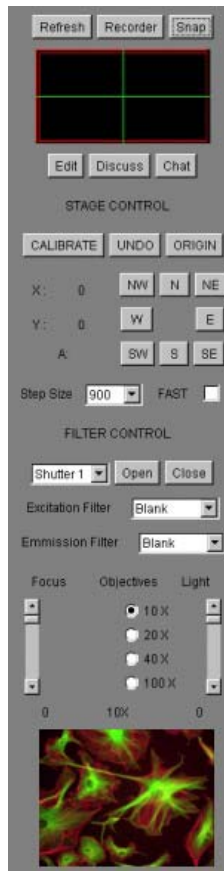
The automatic refresh feature has been changed to function as a "virtual" <Video Recorder>. The pre-view window shows the images captured from the camera at a rate specified by the user. The default rate is one frame/second. This rate and/or delay can be changed in accordance with the 'Clients' request. This feature is additional to manual refresh function that updates / grabs a new image from the microscope when "called" to do so.

A panel has been added to the 'Telemic Client' to facilitate control of illumination intensity and magnification (10X-100X objective) change. This module will be completed and implemented when the requisite supporting hardware is installed.

A <Chat> function is fully operational, allowing on-line users to converse with other Telemic subscribers. This chat feature allows the users to also have private "conversation" with pre-selected users. One can also change nicknames during chatting as well as access the on-line help system.

In the <Discuss Mode> users can discuss a captured image among each other. In addition, a dynamic pointer is invoked such that it can be "surrendered" to any one of the user in the <Discuss Mode>. If a user moves the arrow around and repositions it then a corresponding arrow on any of the active screens will be repositioned accordingly. <Tiling feature> allows a user to create a montage (topographic map) of a specific region of the culture. This feature will allow automatic control of the stage to facilitate capturing "tiles" of the area and then to piecing them all together.





**Figure 11: Cellular Observatory Remote-Access Graphical User Interface**

## **Part II—Cellular Network Dynamics**

### **II A - Engineered Cellular Cultures:**

We began the development of functional assays for observing and detecting the formation (making and breaking) of inter-cellular (junctional) communication in engineered cardiac tissue cultures. We are operating under the assumption that the relative increase in gap junction assembly is indicative of the integrative “cooperative” cellular networks, i.e., why cells form bonds or channels of communication that don’t do anything.

The motivation for this line of thinking is in part related to the observation that the “program”, not to re-regenerate its cellular network structure, limits the recovery of a damaged heart muscle. One approach to address this problem is to repopulate the impaired myocardium by healthy cells. In particular, our live-cell ‘Observatory’ provides an ideal platform for studying the role that the extracellular microenvironment plays in the “assimilation” of non-cardiac cell types such as fibroblasts, skeletal myoblasts and adult stem (hematopoietic) cells. For example, observing the process of stem cell-cardiomyocytes dynamic interaction (differentiation) in an ex-vivo setting may give clues for improving their integration within the intact cardiac cellular architecture.

Two new studies that will exploit the capabilities of the cellular observatory platform are summarized below:

### Myocardial Tissue Engineering and Regeneration Using Adult Stem Cells

Cardiac function is mechanistically linked to junctional cell-cell contacts that give rise to synchronous electrical activity and coordinated mechanical contraction. In contrast, it has been observed that the ability of tumor cells to proliferate is inversely related to its number of intercellular junctions that they form. Dr. Peter Leikes (School of Biomedical Engineering) recently joined forces with us to pursue the development of stem cell technology as a source of cardiac precursor cells in assessing and optimizing cardiac differentiation using their respective expertise (in cellular/molecular tissue engineering, computer-aided microscopy/live-cell observatory and computational modeling.) Dr. Sunil Rangappa (Cardiothoracic Research Fellow) joined our group in July to work along with Dr. Hilmi Ege, (Oncology/Hematology Visiting Fellow) on the tissue culture and genetic engineering strategies designed to promote cell proliferation and growth.

### Myocardial Tissue Engineering and Regeneration Using Electrically Conductive Polymeric Scaffolds

In this study undertaken in collaboration with Dr. Peter Leikes (School of Biomedical Engineering) and Dr. Yen Wei (Chemistry), we intend to utilize the capabilities of the live-cell microscopy system to assess the functional attributes of the ensuing “cardiac syncythium”, growing on electrically conductive polymers.

Specifically, we will

- visualize, quantitatively analyze and model the topology of beating patterns
- assess the establishment of intercellular connectivity, using lucifer yellow as marker for cell-cell communication
- record the electrical activities using multi-array electrode chambers.

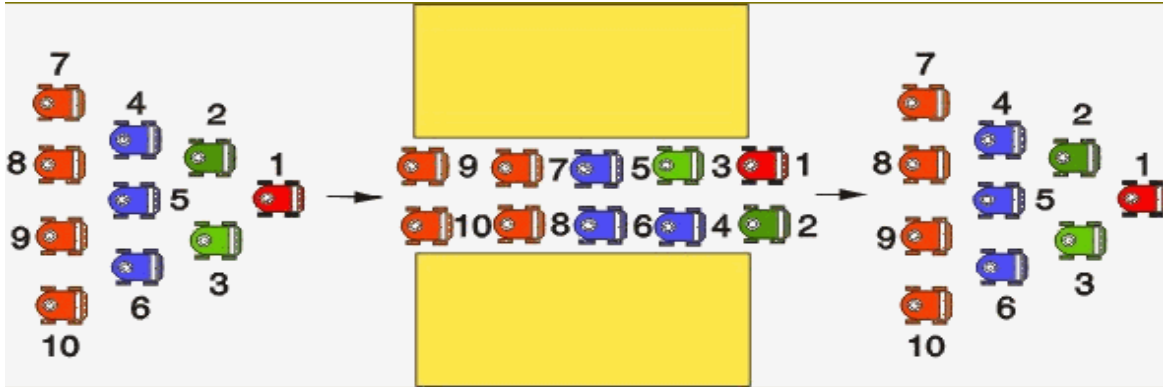
### **II B - Computational Models:**

The imaging utilities of the cellular observatory are aimed to monitor cellular network dynamics, in particular, live-cell spatio-temporal organization and the role of intercellular communication and its effects on tissue genesis, viability, adaptability and survival. Computational modeling of cellular integration and tissue organization dynamics are critical steps in understanding the unfolding and emergence of “organotypic” function.

It has become apparent that the computational modeling objectives of the project are compatible with Dr. Jaydev Desai’s (Mechanical Engineering and Mechanics) interest in the dynamics and control of robotic assemblies as summarized below. He was invited to join the biocomplexity modeling team currently consisting of Dr. Onaral, Dr. Kresh and Mr. Can Evren Yarman, a graduate student who joined the project in March 2001. Mr. Yarman is provided assistance Dr. Desai’s project in order to learn the graph theoretic approach employed in the study of robotic

populations. In parallel, he investigated other public domain ‘complexity’ tools amenable to capturing key features of cellular dynamics.

The objective of the Mobile Robotic Assemblies effort is to develop novel strategies for controlling and changing formations of mobile robotic agents using sensory information and abstract computational models. Initially, the robots rely on vision and sonar systems for tracking other robots, pose estimation and obstacle avoidance. The problem of maintaining the shape of the formation is done through classical non-linear control techniques. Algorithms for transitioning from one formation to the other are developed using graph theoretic techniques.



**Figure 12: Formation of robots changing shapes**

Teams of robots are modeled in formation in terms of parameters  $(g; r; H)$ , where  $g \in SE(N)$  represents the gross position and orientation of the lead robot in that team in  $N$  dimensions ( $N$  equals two or three),  $r$  is a set of shape variables that describe the relative positions of the robotic agents in the team, and  $H$  is a control graph which describes the control strategy (or behavior) used by each robot, and the dependence of its trajectory on that of one or more of its neighbors.

Since “biocomplexity” inspired principles of self-assembly and regulation can be explored “in-silico,” we view this modeling phase as a first step toward addressing the long-term objective of the project directed at gleaning (reverse engineering) biological principles likely to inspire the design of robust and adaptive communication networks as well as other potential applications of interest to DoD. Mimicking biological organization and function are particularly intriguing since they may have implications in engineering distributed memory systems, robotic assemblies and communication networks which exhibit emergent behavior optimized by evolutionary / adaptive processes. These concepts may in turn be generalized to implement next-generation autonomous agents; cellular automata; “flocking” self-organized / coalition behavior; evolutionary pattern search and optimization algorithms; spatial genetic algorithms; and dynamic percolation networks.

This work has made excellent progress. The completion of the installation of the 1Gbps free-space optical link between the Drexel and MCP Hahnemann campuses will enabled the demonstration of this collaboration-enabling capability over an metropolitan area network, and ultimately over a wide area network.

We developed the code to communicate, control and capture images from the cooled CCD monochrome camera (Photometrics CoolSnap FX). A dynamic link library has been created that can be called and used by Java enabled code. The functionality provided by this dynamic link library has been incorporated into the existing Telemic 'Server', enabling it to take control of the camera and capture images from it. This was a critical and necessary step and it is important to recognize that this is one of the only uses of this camera in a remote-controlled environment. A prototype 'Client' was developed to test 'Client-Server' communications and image acquisition from this particular camera.

While capturing images a client can control the following features of acquired images:

Exposure time (milliseconds - seconds): Length of time the charge coupled device (CCD) is accumulating charge. This is the time interval for which the charge coupled device (CCD) is actually exposed to light excitation.

'Binning' A method of increasing or multiplying the light-collecting area on the charge-coupled device by combining the charge from adjacent pixels so that the total charge can be read out as an image. Horizontal and Vertical binning parameters can be set.

Gain (defines the relationship between the number of electrons acquired on the CCD and the analog-digital units (ADUs) generated.) This feature allows the intensity levels in the captured image data to be increased to yield brighter images.

CCD capture region. A region is a user-defined, rectangular exposure area on the CCD. By specifying 4 co-ordinates the 'Client' can control what array of the charge coupled device (CCD) should be captured. This feature allows one to capture images of a desired region from the entire field of view.

Image Format. Images can be captured in a number of standard file formats

- a) 8 bit JPEG
- b) 8 bit TIFF
- c) 16 bit TIFF

Time Lapsed Image Acquisition: A 'Client' can specify the number of images required and the delay or time lapse between images (seconds, minutes, or hours.)

Color Rendering ('Client' side): This feature also known as 'Pseudo Coloring' is under development and refinement. A captured image can be pseudo colored by applying a color spread of red, green, blue, magenta, cyan and yellow. This feature is particularly useful for capturing image data at different wavelengths and creating a composite of these images.

### **Enhancements and Improvements**

The ability to capture images from the Firewire camera and write them to disk has been somewhat cumbersome because of the requisite intermediate steps necessitating first writing the

captured data to disk as a 'bitmap' and then taking this 'bitmap' image and converting it to a 'jpeg' format and thereafter making it available to the client.

This feature has been considerably enhanced. Now there is no intermediate creation of bitmap images and 'bitmap' to 'jpeg' conversion. Images can be captured and written to disk directly as true 'jpeg' files. A feature that is critical for the high-speed image capture and transmission.

At present with existing hardware and networking (using 100Mb/s Ethernet card) infrastructure, an image capture rate of 16-18 images per second can be obtained while concurrently writing the images to disk. Thus the client can access images from a buffer on the server thus enabling a video capture rate of ~15 frames per second, with slight breaks or stops interposed to allow for buffering of data on the 'Client'.

The Telemicroscopy software has undergone several changes over the course of this research – the most important being the completion of the integration of its various modules and versions into one, composite, software. The Telemicroscopy 'Client' and 'Server' now feature full control of the Microscope and both the Firewire and Coolsnap cameras. The 'Client' can now either use the Microscope with the CoolSnapFX camera or the Firewire camera. Use of both cameras at the same time is possible but is constrained by the optics of the microscope design.

The Firewire camera serves real time video from the microscope. Video rates of up to 10 frames/sec can be obtained. This functionality is now available through the 'Recorder' button on the Microscope Control Panel. A separate Server has been written for this function. All functions can still be accessed using the same Telemic Client.

### **CoolSnapFX camera**

Several new and important functions have been added

- a) Tiling: This feature allows the 'Client' to generate a Topographic Scan of the Specimen under investigation. Here the user can map out a specific region of the slide. This area will be not viewable under one view from the camera. Hence one can program the stage for capturing pieces of that area and then to piece them all together.
- b) Image Saving: Images obtained from the microscope can now be saved. A file Server and Client have been developed to provide this functionality. Images are saved on the Server Side. The images saved on the Server by the Client can be previewed and saved to Disk on the Client Side. The Client can preview the saved images and download selected images in a zip file.
- c) Image Processing: Several more Pseudo Coloring options were added along with some new Image Processing functions for captured Images.

### **Microscope Control**

- a) Objective Control: Objective changing is now possible by means of a newly acquired turret changer module. The client can change between 4x, 10x, 20x and 40x objectives.

- b) Focus Tracking: This feature allows the client to focus back to a snapped image. Thus a 'Client' can go back and forth between preferred focus settings.
- c) Access Control: This feature allows only one user to take control of the microscope and cameras. The control can be released and requested. The control is circulated among 'Clients' as a token. Functionality other than microscope and camera control, like Discuss, Chat, etc is available to passive clients.

This work was demonstrated, via remote access to the system, at the Next Generation Internet Principal Investigator meeting on January 6, 2002 in Tyson's Corner VA.

### **Telemicroscopy Software version 2.0**

The current release of the software has reached the designation of version 2.0

The telemicroscopy software or 'Telemic' has undergone several noteworthy revisions, which include a complete redesign of its GUI (graphical user interface) and added new and critical features. Several persistent bugs have also been eliminated, the most important being the crashing of the 'file server' after several hours of online activity.

The modifications / upgrades to the software can be classified into three categories.

- a) Changes to Microscope Control
- b) GUI changes.
- c) Bug fixes

### **Changes to Microscope Control**

The ability to move the stage to any specified X-Y coordinates, within the available field of excursion is graphically emulated on the screen that maps the area available for stage movement and then provides cursors to mark the x-y coordinates.

This graphical emulation or the 'stage movement area' has been redesigned to dynamically depict the current position of the stage within the available field of excursion. When a user connects to the 'Telemic' the crosshair will move to reflect the current position of the X-Y stage in the entire field of movement for the X-Y stage. This initial position will be the defined 'origin' for the users current session. The 'origin button' will bring the user back to this point where the user's session started. The origin button will not move the stage to the origin of the stage. To accomplish this one will have to use the 'calibrate' button. The crosshair will always reflect the current position of the stage within the field of movement after both coarse and incremental movements of the stage.

Full 'UNDO' functionality is now provided to undo both incremental and coarse movements of the stage.

A new and much awaited function has been added which allows the users to mark particular positions of interest within the field of movement of the X-Y stage and retrace back dynamically to these positions. Currently a user can mark a total of 8 positions. The marked positions are displayed with a numerical "stamp" on the stage movement screen. To retrace back to a marked

position the user can select the appropriate position number from a checkbox provided on the microscope control panel.

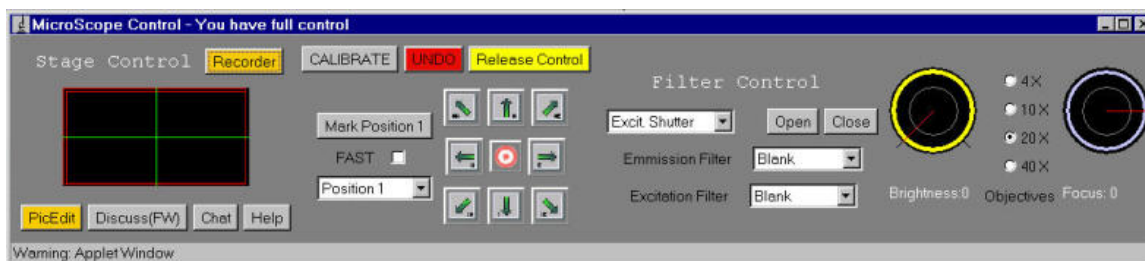
## GUI Changes

Several GUI (graphical user interface) changes have been made to the Telemic ‘Client’ which include repositioning, addition/deletion of several controls. The most important among these changes being, the redesign of the illumination and focus control to a round-knob slider from a scrollbar.

Most importantly, the focus control knob can be rotated, by dragging the mouse, either in the clockwise or anti-clockwise direction to increase, decrease the focus respectively. This enables affectively an infinite bi-directional loop (no fixed end point restriction!).

The illumination or brightness control knob can be rotated to set brightness (light intensity levels) between 0 and 10V DC.

It is important to emphasize that both these round-knob slider controls provide true real-time control of their respective hardware on the microscope.



**Figure 13: Improved GUI**

## Bug Fixes

- 1) Correction of instability in the File Server. The file server memory management functionality has been redesigned. This corrects a problem that caused it to crash after several hours of usage.
- 2) The “collaboratory” discuss feature has been reworked to eliminate mouse sensitivity problems, and has been made more intuitive.

The changes/new features summarized above have been implemented after many sessions of use, and incorporate end-user feedback / critique. These “test” sessions were the major motivating factor for the Telemicroscopy software upgrade from its earlier incarnation (beta) releases to a more robust and fully integrated version 2.0.

The development process of the Telemicroscopy software is now complete. The software is fully operational and has been thoroughly tested on High Speed Network and the Drexel 100 Mbps network.

## **New Horizons**

### Sony EVI-D100 Web camera:

We experimented with adding live net-conferencing capability using a Sony EVI-D100 camera that is specially designed to allow multiple degrees of freedom control. This camera has a programmable pan, tilt, zoom, focus and several modalities of image capturing and previewing. We have incorporated this camera into the Telemic project to serve as a “third eye” by customizing our client-server based software. Accessing this camera and viewing live images provides the “remote” participants with added flexibility for panoramic observation of distant events of perceived telepresence.

The camera is interfaced to a frame grabber card (IEEE-1490 Matrox Meteor-II), allowing the end- user to capture and preview real-time video. The camera control system is accessed via its RS232 (serial port) interface to the server. The camera ‘Server’ interface is Java enabled, allowing control of various camera functions such as: pan / tilt / zoom, etc. A retrace capability is being added to facilitate recall (up to 8 positions).

### Multi-Channel Remote Data Acquisition System:

It became apparent to us that imaging cellular culture evolution over time will be greatly aided by the ability to monitor function attributes of the cellular network dynamics. In particular remote multi-channel (real-time) data acquisition would be very useful.

Signal acquisition is an integral part of all experiments conducted using tissue culture and cellular engineering strategies to promote cellular proliferation and trans-differentiation.

In view of the above, the proof of concept work has been started with the goal of integrating a data acquisition capability into the Telemic. We are testing this concept, which is a 32-channel waveform recording system (A/D 16-bit measurement resolution, 250kHz sampling rate, and a maximum measurement range of  $\pm 10V$  full scale). This device is interfaced to the PC via a standard USB port.

We have run several simulations in a networked environment. The current prototype enables real-time data 16-channel signal acquisition, and remote display (no remote control).

A dedicated browser can be used to preview recorded waveforms in real time, limited only by network speed and server disk access time. The tissue culture and cellular engineering studies conducted using this extension of the Telemicroscopy software consisted of a programmable (4-channel) stimulator that paced the cells for several hours. We are currently working on developing a client server model for interfacing a commercial data acquisition device (Windaq DI-720) with the Telemicroscopy software using Java.

### High-speed Network Implementation:

We have successfully conducted the testing of the Telemicroscopy software on the High Speed Network. The high-speed network is operating at full Gigabit per second capability. We are currently in the process of upgrading our network interface cards (see specs bellow) to facilitate multi-platform (Sun/Solaris 8; Windows NT; Windows 2000; Wireless 801a protocol) imaging and signal acquisition from the live-cell experiments.



This card provides easy, cost-effective migration to Gigabit Ethernet over Category 5 cabling.

High network performance and flexibility, using PCI-X bus at 64bit/133MHz alleviates server bottlenecks while maximizing uptime.

Link types supported: Ethernet 10Base-T, Ethernet 100Base-TX, Ethernet 1000Base-T  
Standards compliance: IEEE 802.3-LAN; IEEE 802.3U-LAN; IEEE 802.3z -LAN; IEEE 802.1Q; IEEE 802.3ab-LAN; IEEE 802.1p Plug and Play.

### **Live Cell Microscopy- Summary of Results and Lessons Learned**

A collaborative computational environment was created to enable topographic data acquisition and processing remotely. The microscope system that we implemented is well suited for remote control, and is equipped with direct digital image recording and computer interfaces to the microscope optics and cell culture stage. Importantly, the interactive control of image acquisition is achieved from any site on the Internet. Sophisticated image analysis and visualization is also provided for those accessing the microscope from a remote site. It is quite feasible that tasks requiring extensive computation can be “transparently” distributed to high performance computers on the network. The goal of porting the associated image processing and analysis software into a network-based resource available to geographically distributed researchers was fully realized.

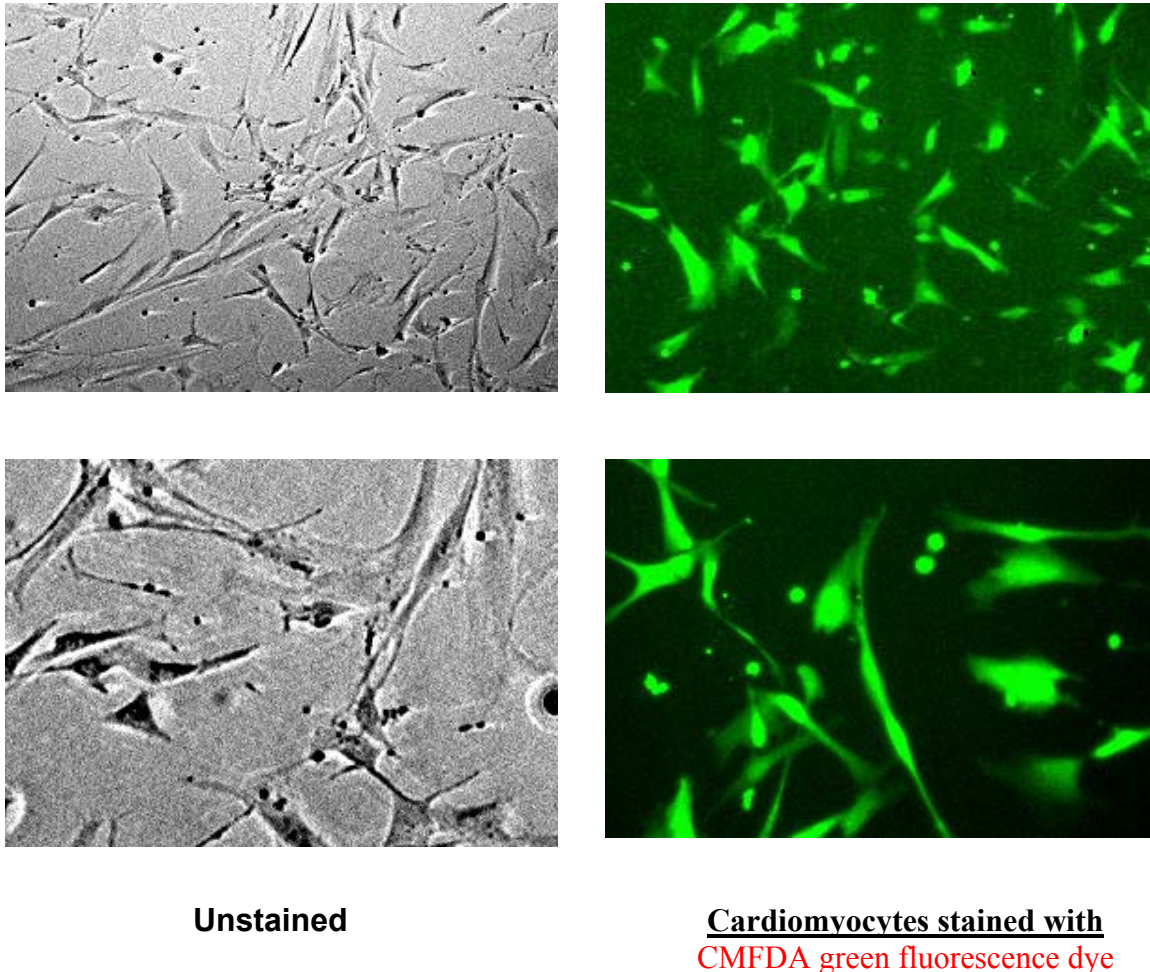
The ability to acquire images remotely (and control the microscope stage movement) facilitated the study of the electro-mechanical perturbations on the dynamic behavior of fluorescent-labeled structures across large spans of cellular terrain that would not be possible otherwise

**Live-cell microscopy isn't a simple computer program.** To do good (2D / 3D) microscopy, you really do need to understand optics, statistics, digitization and their interactions. There is always a tradeoff between getting enough photons from your specimen to make a good image and hurting the specimen (maintaining cell viability) by subjecting it to too much illumination. Doing good live-cell microscopy implies making every effort to optimize the optical and digitizing conditions.

Because the correlation between in vitro and in vivo phenomena is paramount for mammalian live-cell biology, it is of the utmost importance to accurately simulate the host conditions of the isolated specimen during live-cell microscopy. One solution to the above problems is to observe cells in culture maintained using a micro-environmental control system increasing the complexity (with justification) of Telemic accessibility.

*Cell Biology 2001*, a market research study just released by Microscopy/Marketing & Education (MME) reveals that two thirds of the microscopists surveyed at December's American Society for Cell Biology (Washington, DC) currently use or, within the next year, plan to move to live cells as the basis for their research. Prompted by the recent introduction of vital fluorochromes such as GFP (green fluorescence protein), this new paradigm has far-reaching implications for fundamental microscopy as well as the fields of drug discovery and biotechnology. The inevitable desirability of distributed cell imaging capability is a natural progression of the ever-

increasing network bandwidth and need for multidisciplinary collaboration and information sharing.



**Figure 14: Live adult human (mesenchymal) stem cells co-cultured with human cardiac myocytes**

### **d.3 Gbps Infrastructure (Stewart Personick, et. al.; Dom Imbesi, et. al.)**

The fiber link between Drexel and Penn was installed, and is operating at net-payload throughputs exceeding 200 Mbps in support of our distributed neuro-anatomical distributed database access activity. This net-payload TCP/IP throughput is exceeding our expectations for off-the-shelf PC-to-server performance.

We (Drexel and Lucent) installed a 1-Gbps free-space optical link at sites on the Drexel and Hahnemann University. The free space optical equipment is transmitting 27dBm of optical

power and receiving -14dBm in each direction. Power margins are more than adequate to ensure error free bi-directional transmission at the Gbps rate. The installation, integration and test activity was conducted over a two-day period. Some reliability problems were encountered, and were corrected.

#### **d.4 Global Digital Mapping (Somerset Geographics: Brian Schmult)**

##### **Assessment of Worldwide Data Availability to Support Real-time Fly-Through**

###### **The Objective:**

The overall objective of the research is the creation of a compelling application that requires high-bandwidth communication, given an appropriate set of assumptions about the computing environment. The application under consideration is the interactive fly-through of real-world terrain with a complete world model available for queries and analysis, plus on-the-fly simulations of events (forest fires, precipitation runoff, etc.) This combination of requirements is important, otherwise a user will just download some data once and this is no longer a communications problem. The real-world terrain requires streaming imagery, aerial or satellite. For access to arbitrary locations, this is probably too large to store locally. The simulations imply the need to do processing locally, possibly on the imagery, thus requiring the raw data. (Otherwise this is simple rendering that might be done at a server followed by MPEG compression, greatly reducing the communications requirements.) Finally, the model is critical, as strictly image data is of little value.

Basic fly-through using imagery and vector data is easy and has already been done using locally stored data. The key to a more compelling application is in the completeness and accuracy of the dataset. (i.e., the vector roads must overlay those in the image.) The completeness is also important to ensure that communications remains important. If the data is spotty people will download their area of interest once and save it. The ability to look anywhere will keep people looking everywhere and make local storage impossible.

###### **Methodology:**

References to non-classified data (commercial and free) are now readily available on numerous Internet sites. Many sites have pages devoted solely to links to a multitude of other sites housing specific data. Hence the main initial strategy was to search for the specific data required. To the extent that it is already available, portions of the activity are complete with only the organization, assessment and documentation of the quality of said data. Filling in the holes then requires more research. The methodology for this research was not developed adequately for a write-up at the time of termination of this activity.

###### **Required Data:**

These are the general data requirements:

1. Aerial and/or satellite imagery, as consistent as possible.
2. Digital elevation models, as consistent as possible.
3. Roads, political boundaries, significant hydrology and possibly railroads. Must be precisely located so that they sit properly on the imagery.

4. Building locations as centroids, rough and/or detailed outlines. These are not strictly necessary, but would greatly enhance the utility and appeal.
5. Attributes: feature names, building addresses, etc.

In all cases the objective is to locate a dataset that is as broad and self-consistent as possible. In practice this will be done on a national and regional basis. Data from different regions will not be of the same quality, and may not edge-match.

### **Imagery:**

Imagery is collected by satellite or plane (aerial). Traditionally satellite coverage is much broader but at much lower resolution. More recently higher-resolution satellites have become available. Their resolution is good enough for our purposes, so it may be only a matter of time and money to get the coverage. Aerial imagery can be of much higher resolution, depending on the requirements and cost of the mission. Coverage can be spotty especially for very high-resolution missions, which are necessarily flown at low altitude and hence are time consuming. Note that at least in the U.S., the NAPP (National Aerial Photography Program) provides nation-wide coverage with consistent attributes, except that the collection time (year) is spread out.

### Attributes of Imagery

The most basic attribute is the resolution. Two definitions are used by the United States Geodesic Survey, USGS. The first is spatial resolution, the minimum distance between two adjacent features or the minimum size of a feature that can be detected. The second is the ground sample distance, GSD, the distance on the ground represented by each pixel in the x and y components. The spatial resolution is usually greater (of lower quality) than the GSD. GSD is the figure usually quoted.

The other primary attribute is spectral content. Generally an image will be Black & White, BW, Color, or Color Infrared, CIR, CIR is also called pseudo color and false color. In a CIR image the normal visible colors are used but they do not correspond to visible colors. Rather vegetation has one color, asphalt has another, etc. Satellites can also use non-film sensors to record at other wavelengths.

Imagery is distributed in chunks whose size and extent depends on the resolution. Two common sizes in the US are the quadrangle and the quarter quadrangle or quarter-quad. The quadrangle is a rectangle 7.5 minutes in latitude by 7.5 minutes in longitude. (Note that it is rectangular only when projected into a planar coordinate system.) A quarter-quad is 3.75 minutes on a side.

Raw images contain both camera distortion and elevation distortion. Elevation distortion means that the top of a tall object located anywhere except along the camera axis will not be properly located. Instead of being directly over its base it will be off to the side. This means you cannot determine its coordinates. This can be automatically corrected in the case of terrain by applying a digital elevation model. This permits the pixels to be moved to their proper place. There is presently no cure for non-terrain objects, such as the Washington Monument. Images corrected in this manner are known as “orthophotos.” The same process used to create orthophotos also corrects for camera distortion.

Coverage refers to how much of the total world is available. This is different from vector data since vector data can cover empty areas (i.e., water) properly. Imagery however shows up as missing unless the area is explicitly indicated to be water. In such cases software can fill in such spaces on the fly, or images can be generated that match adjacent existing images.

### United States Imagery

Imagery for almost the entire country is available from the United States Geological Survey (USGS) through the National Digital Orthophoto Program (NDOP), whose primary source of imagery is the National Aerial Photography Program (NAPP.) This is the same data served by the well-known Microsoft TerraServer. These images are distributed as Digital Ortho Quarter Quads (DOQQs.) The main Web site at the USGS for DOQQs is

<http://www-wmc.wr.usgs.gov/doq/>

Among other things, complete specifications are available in PDF files for downloading.

The GSD resolution of a USGS DOQQ is 1 meter. They can be either BW or CIR. According to the USGS Web site, complete DOQQ coverage of the conterminous United States is expected by 2004. Thereafter the update cycle will be 10 years for most areas, and 5 years in areas where land use change is more rapid. The above Web site has further details on the current coverage status. The people at USGS-EROS say however that the state maps on that site are not always up to date. A current data file is available at

<ftp://edcftp.cr.usgs.gov/pub/metadata/DOQ/doqq.tar.gz>

The US coverage by DOQQs is presented via table and map. The USGS data files do not describe quarter quad areas that are not yet done, nor ones covering only internal water bodies. That data was derived by comparing the completed areas against a coarse resolution map of the conterminous US. This gives an approximate count of water areas and of incomplete quarter quads. The table is shown below. It is apparent that coverage is available only in B&W form. By converting areas of CIR-only coverage to B&W, we can get 88% coverage of the conterminous U.S.

Type of Quarter Quad	Count	% Land
Total Land Quarter Quads	213,027	
B&W Quarter Quads	152,883	72%
CIR quarter Quads	44,938	21%
Available as either	187,585	88%
Missing	25,442	12%

Table 2: US Coverage by DOQQ

It is possible that holes in populated areas might already be filled by local acquisition. This is relatively unlikely however since the NAPP concentrates on populated areas. The holes are hence unlikely to be of interest to private parties, or to local governments who would have to foot the entire cost of a mission.

The latest satellites can create B&W images comparable to those from the NDOP. Space Imaging's "Pro" series has a 1-meter GSD pixel and an RMS error of 4.8 meters, and is supposed to meet U.S. National Map Accuracy Standards for quarter quad mapping. These are also ortho rectified. Hence they could in theory fit with the DOQQs.

However, at retail prices it would cost in the tens of millions of dollars to collect the roughly 25,000 quarter quads still missing. It is unclear whether this would be feasible, since the data is apparently going to get collected by the USGS anyway.

### **Opportunities for future work:**

Image, DEM and vector data for the US is generally available from the Federal Government, although in various forms. Many Government and other web sites will contain pointers to data sources. Holes in national coverage will be more difficult to fill, as legwork will be required to see if suitable local datasets are available. Building data is not yet generally available although research is ongoing on automatic extraction from imagery. Other national data should be pursued in a similar manner, starting with national government sources in each country, plus regional suppliers. There are several web sites devoted to regional data, plus lists of national suppliers. <http://harbert.geology.pitt.edu/images/web.html> appears to be one of the better ones. After this, substantial effort will be required to fill in holes.

### **d.5 MONET East Ring (Lucent)**

Ongoing support was provided by Lucent for the maintenance of the East Ring of the MONET optical networking testbed.

## E. All-Optical Networking Devices and Subsystems

### e.1 All-Optical Networking Devices and Subsystems (Paul Prucnal, et. al.)

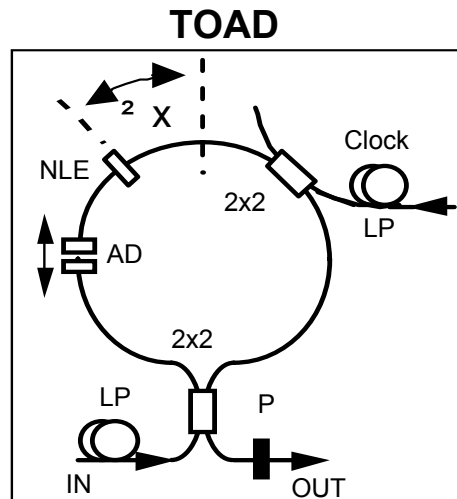
This research is focused on the demonstration of an experimental prototype of an all-optical digital regenerator that is of the “3R” type” : i.e., including functionality for reshaping, retiming, and regeneration. The regenerator is based on a device called a “TOAD” (Terahertz Optical Asymmetrical Demultiplexer).

#### **Optical pulse width management / format converter**

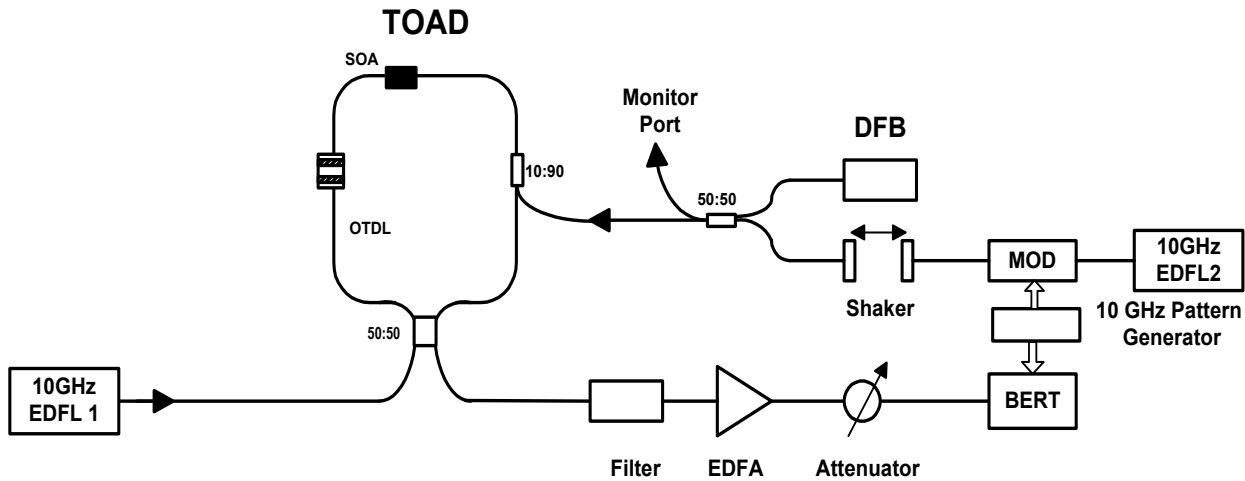
An optical pulse width management / format converter was designed and demonstrated. The format converter is made of an optical Sagnac loop (interferometer) with a semiconductor optical amplifier, SOA, in the loop, displaced from its center. The degree of displacement of the SOA from the center of the loop determines the output pulse width. The format converter can convert a return-to-zero, RZ, formatted signal to non-return-to-zero, NRZ, format. With its setup, one can stretch 1.3ps input pulses to any value between 4ps to 58ps. This means that, for bit rates above 20Gb/s (50 ps pulse spacing), one can convert RZ formatted data into NRZ formatted data. For bit rates lower than 20Gb/s, the setup provides a powerful tool to manage pulse width.

#### **Experiments to test a method for reducing data timing jitter, for use in our proposed 3R regenerator**

The idea underlying the experiments described below is to send OC192 (10 Mbps) data into the “Clock” port of a TOAD, and to use this data to switch out clock pulses entering the “IN” data port of the TOAD.



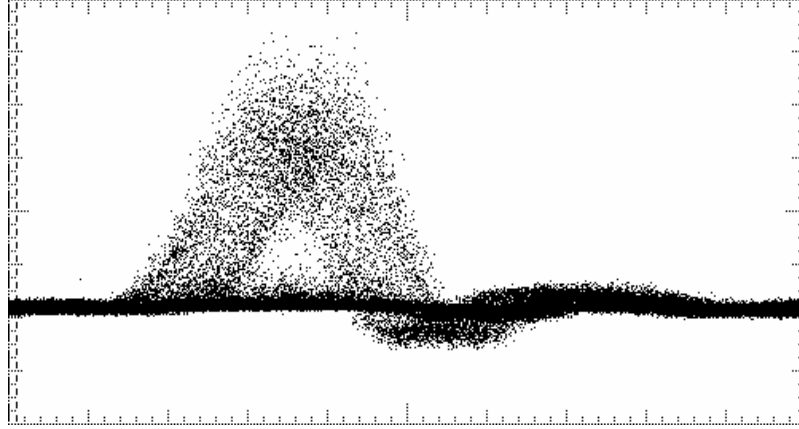
**Figure 15: TOAD Schematic**



**Figure 16: Experimental Setup for Timing Jitter Reduction**

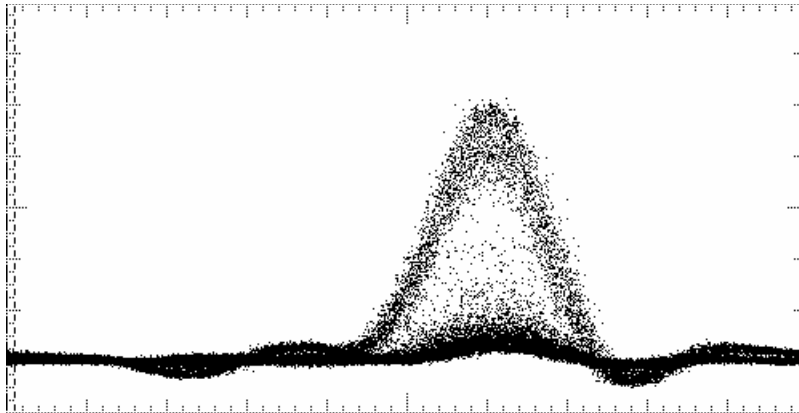
If we open the TOAD's switching window wide enough, we can then accommodate timing jitter in the data (entering the TOAD clock port) that is less than the switching window i.e., we can still switch out all of the clock (entering the TOAD data port). The optical signal that is switched out will have all the information (i.e., pulse on / pulse off) that the input data has, but with no timing jitter. In addition, because of the TOAD's transfer function, typical of most interferometers, amplitude noise from the input is reduced after going through the TOAD. This is because the intensity fluctuations at the input become phase fluctuations in the TOAD. And since we're operating near the PI phase shift region, the output varies very little with respect to the phase shift. An inherent limitation of this configuration is its pattern dependency... due to the finite recovery time of the semiconductor optical amplifier, SOA. This effect is more evident at 10Gb/s than 2.5Gb/s. Injecting CW light of a different wavelength into the SOA can reduce the recovery time of the SOA, thus reducing the pattern dependency. The experimental setup we used to demonstrate this is as shown in Figure 16. An improvement in performance, using CW light injection, was demonstrated, initially, using eye diagrams.





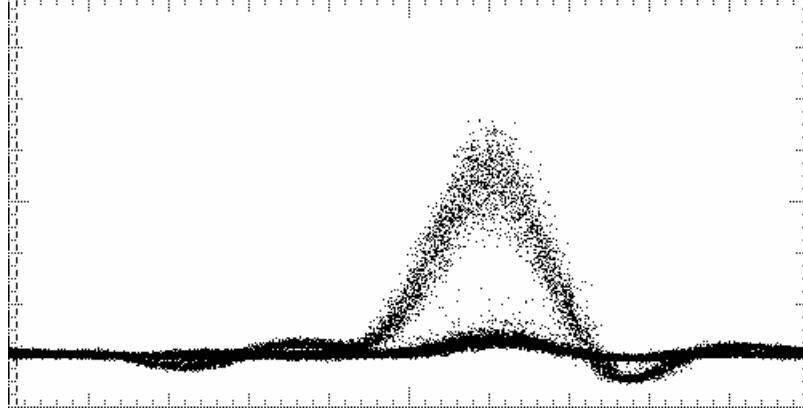
**Figure 17: Input to the TOAD, with added timing jitter**

In Figure 17, timing jitter has been added to the input data stream by increasing and decreasing the length of an optical path (“Shaker” in Figure 16) using a mirror on a speaker operating at 10 Hz. The timing jitter is 9 ps.



**Figure 18: The output of the TOAD, when we inject a third wavelength of CW light into the TOAD**

In Figure 18, CW light at 1539nm. We see that the eye is starting to open. This is because the CW light is reducing the carrier recovery time inside of the SOA.

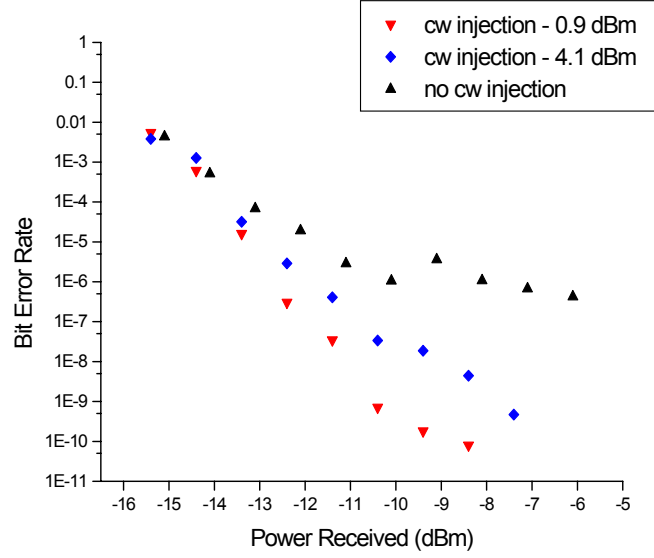


**Figure 19: The output after the TOAD, with the CW bias light level set at: +7dBm**

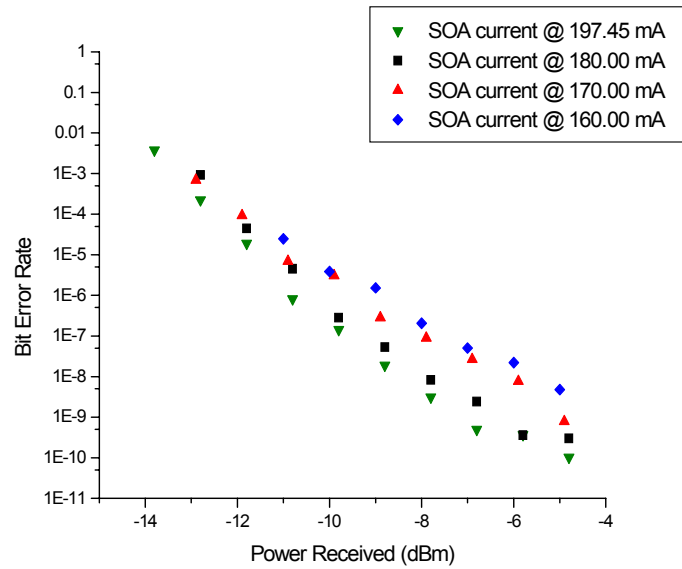
In figure 19 the eye is wide open. We see that the amplitude variation is also decreased slightly.

The improvement in jitter tolerance that can be obtained using CW light injection can also be demonstrated using bit error rate measurements. Below are bit error rate, BER, curves plotted for different CW injection power and SOA current values.

Shown in figure 20, a higher level of CW injection power leads to a decrease in BER. For no CW injection, a BER floor exists at  $10^{-6}$ . A CW injection of  $-4.1$  dBm eliminates the BER floor at  $10^{-6}$  and BER less than  $10^{-9}$  was achieved. A CW injection of  $-0.9$  dBm shifts the BER further to the left and BER less than  $10^{-10}$  was achieved. Figure 21 shows the BER as a function of SOA current. An increase in SOA current corresponds to an increase in the recovery rate of the SOA and we see a decrease in BER as a result of higher SOA injection current. The observed response of the TOAD to changing CW injection level as well as SOA current level matches those predicted by theory. Increasing the CW injection level increases the SOA recovery rate and thereby decreases the effect of pattern dependency. This is readily observed in the BER curves shown in figure 20. Increasing the current also reduces the recovery time of the SOA and results in a BER curve shift to the left in figure 21. In our experiments done at 10Gb/s, both methods of decreasing the SOA recovery time resulted in a direct performance advantage. At 40Gb/s, we believe that further reducing the SOA recovery time using even higher levels of CW injection is critical for the TOAD to regenerate optical data streams.



**Figure 20: BER vs. CW power**

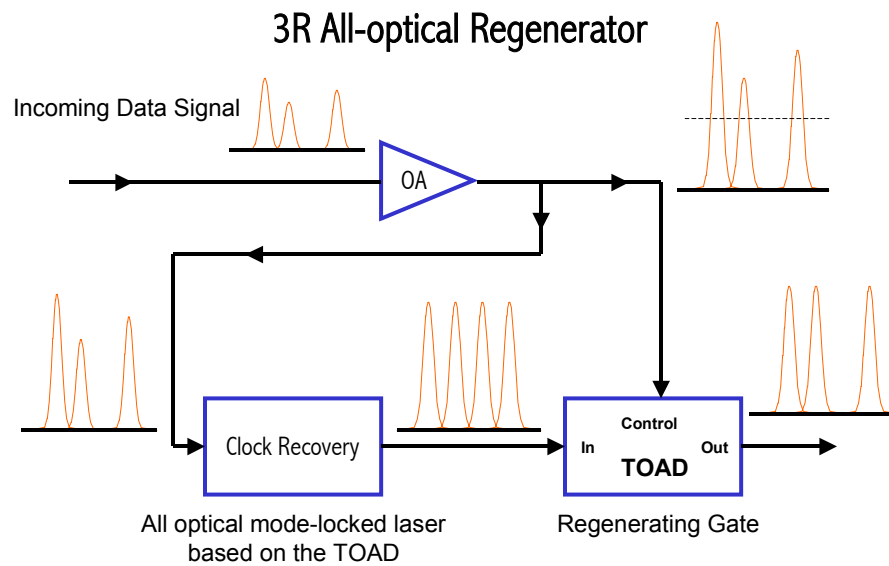


**Figure 21: BER vs. SOA current**

## All-Optical Clock Recovery

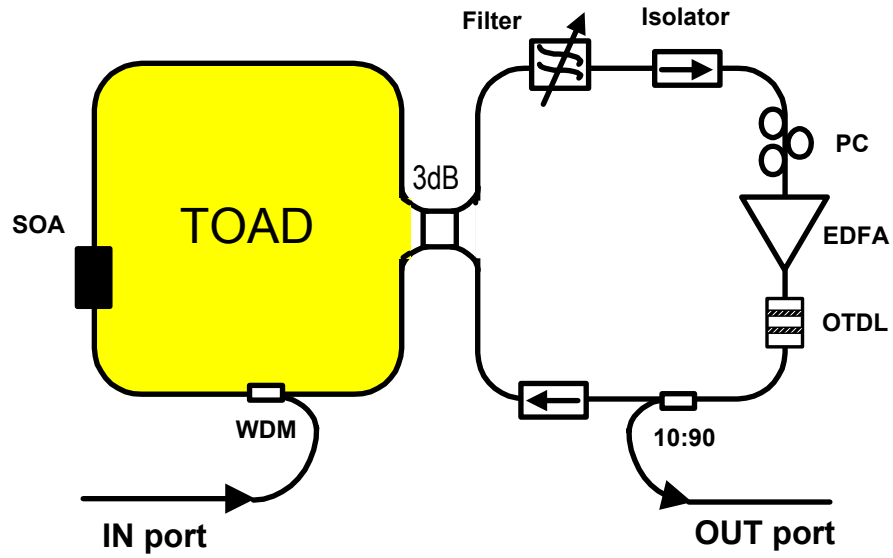
In the above experiments on 3R regeneration we employed an all-optical 3R regenerator that uses a locally generated optical clock. The regenerator shows very high performance, and error free operation at OC-192. However, to make this device more practical, the clock signal needs to be extracted (all-optically) from the incoming data.

Below we explore the possibility of building such a clock recovery unit, based on a mode-locked figure-eight laser with a TOAD as an all-optical gate.



**Figure 22: All-optical 3R Regenerator**

All-optical injection-mode-locking is an attractive way to derive a clock signal from the incoming optical pulse stream. In the mode-locked figure-eight laser, the incoming optical data signal will perform the modulation of the light in the laser cavity. I.e., the TOAD acts as an optical modulator. The proposed experimental setup is shown in figures 22 and 23.



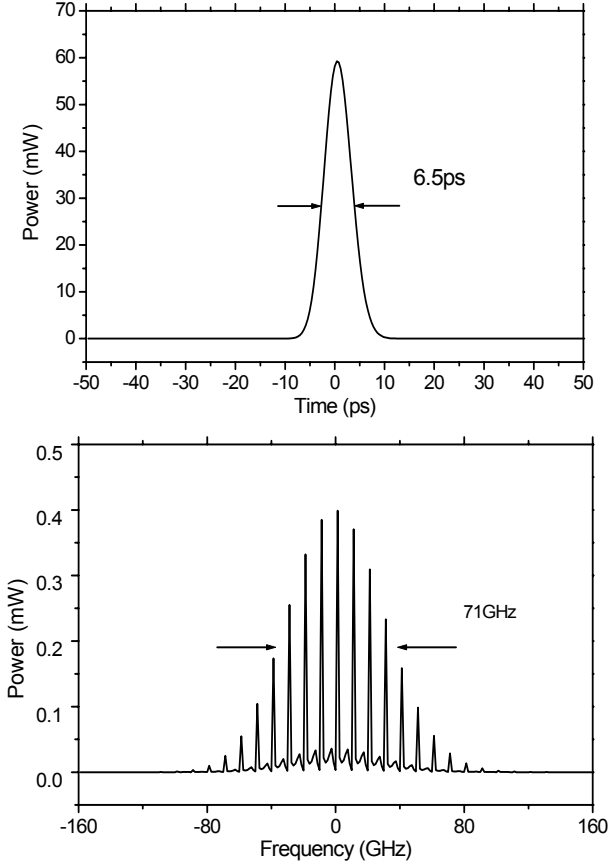
**Figure 23: Experimental structure of mode-locked figure-eight laser using a TOAD**

We have developed an analytical model, with a goal of analyzing the pulse width and spectral characteristics of the recovered clock signal at the output of such a clock recovery unit.

#### **Results obtained:**

Figure 24 shows an example of the simulated output pulse shape and the associated optical spectrum.

The output pulse width (full width at half maximum) is 6.5ps, the spectral width is 71.4GHz, and the time bandwidth product is 0.46.

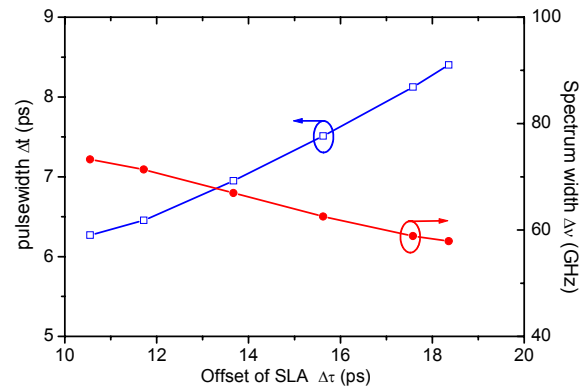
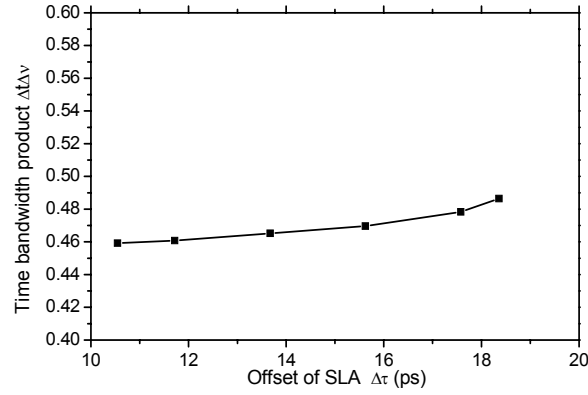


**Figure 24: Waveform and spectrum of the mode-locked output pulses**

In Figure 24 the carrier lifetime is 300ps. The offset of SOA in the TOAD is 12ps. The saturation energy of the SOA is 1pJ. The input pulse width is 10ps. The peak power is 100mW. The bandwidth of the filter is 1nm.

In the mode locking process, the offset of the SOA in the TOAD has a direct influence on the performance of the output clock recovery unit.

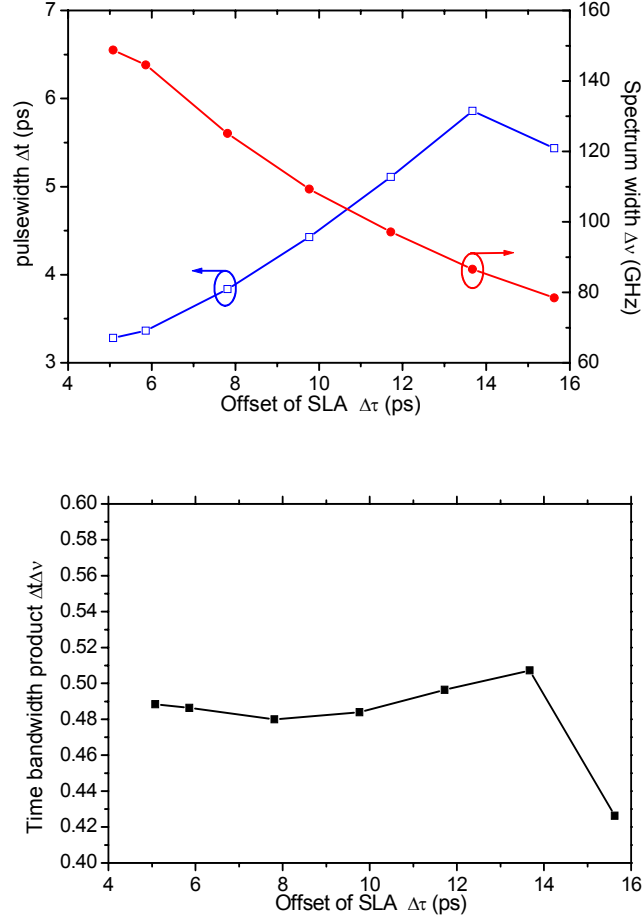
We see this in Figure 25, where, as in Figure 24, the input pulse width is 10ps. The output pulse width is 6-9ps. The time bandwidth product is 0.46-0.48; which approaches the limit (0.44) of Gaussian pulses.



**Figure 25: Pulse width, spectrum, and time-bandwidth product vs. the Offset of the SOA in the TOAD [10 ps input pulses]**

In Figure 25 the input pulse width is 10ps. The peak power is 100mW. The bandwidth of the filter is 1nm. The carrier lifetime is 300ps.

In Figure 26, the input pulse width is 5ps, and the output pulse width is 3-6ps. The chirp of the output pulse increases slightly, with a time bandwidth product of 0.48-0.50. Under these conditions, the output can retain the pulse width of the input.



**Figure 26: Pulse width, spectrum, and time-bandwidth product vs. the Offset of the SOA in the TOAD [5 ps input pulses]**

In Figure 26 the input pulse width is 5ps. The peak power is 100mW. The bandwidth of the filter is 2nm. The carrier lifetime is 300ps.

In both Figure 25 and 26, when we decrease the window size of the TOAD, the pulse width of the output will decrease and the spectral width will increase.

Our analytical modeling shows that a mode-locked figure-eight laser can meet the basic requirements, as an all-optical clock extractor, for an all-optical 3R regenerator.



## REFERENCES

**Note: References below are organized by report sections and reprinted in the APPENDIX, when highlighted in bold font.**

### Section - b.1 Ultra High Capacity IP Router (Martin Zirngibl, et. al.)

- [1] J. Gripp, P. Bernasconi, C. Chan, K. L. Sherman, and M. Zirngibl, **“Demonstration of a 1 Tb/s optical packet switch fabric (80\*12.5 Gb/s), scalable to 128 Tb/s (6400\*20 Gb/s),”** in *Proc. ECOC’00*, 2000, post-deadline paper 2.7. (see appendix page 70)
- [2] J. Gripp, M. Duelk, J. Simsarian, S. Chandrasekhar, P. Bernasconi, A. Bhardwaj, Y. Su, K. Sherman, L. Buhl, E. Laskowski, M. Capuzzo, L. Stulz, M. Zirngibl, O. Laznicka, T. Link, R. Seitz, P. Mayer, and M. Berger, **“Demonstration of a 1.2 Tb/s optical packet switch fabric (32\*40 Gb/s) based on 40 Gb/s burst-mode clock-data-recovery, fast tunable lasers, and a high-performance NxN AWG,”** in *Proc. ECOC’01*, 2001, post-deadline paper ThA4.8. (see appendix page 72)
- [3] J. Gripp, M. Duelk, J. Simsarian, P. Bernasconi, A. Bhardwaj, K. Sherman, K. Dreyer, M. Zirngibl, and O. Laznicka, **“4 x 4 demonstration of a 1.2 Tb/s (32 x 40 Gb/s) optical switch fabric for multi-Tb/s packet routers,”** in *Proc. ECOC’02*, 2002, post-deadline paper PD2.4. (see appendix page 74)
- [4] J. Simsarian, A. Bhardwaj, K. Dreyer, J. Gripp, O. Laznicka, K. Sherman, Y. Su, C. Webb, L. Zhang, and M. Zirngibl, **“A widely tunable laser transmitter with fast, accurate switching between all channel combinations,”** in *Proc. ECOC’02*, 2002, paper 3.3.6. (see appendix page 76)
- [5] A. Bhardwaj, J. Gripp, J. Simsarian, and M. Zirngibl, **“Long-term wavelength switching measurements with random schedules on fast tunable lasers,”** in *Proc. ECOC’02*, 2002, paper 11.5.3. (see appendix page 78)
- [6] [M. Kauer, M. Girault, J. Leuthold, J. Honthaas, O. Pellegrini, C. Goullancourt, and M. Zirngibl, **“16-channel digitally tunable packet switching transmitter with sub-nanosecond switching time,”** in *Proc. ECOC’02*, 2002, paper 3.3.3. (see appendix page 80)
- [7] M. Duelk, J. Gripp, J. Simsarian, A. Bhardwaj, P. Bernasconi, M. Kauer, and M. Zirngibl, **“Next generation packet routers,”** in *Proc. SPIE’02*, vol. 4872, 2002.
- [8] P. Bernasconi, C. Doerr, C. Dragone, M. Cappuzzo, E. Laskowski, and A. Paunescu, **“Large N x N waveguide grating routers,”** *J. Lightwave Technology*, vol. 18, no. 7, pp. 985–991, July 2000.

## Section - C. Architecture

- [9] Personick, S.D. “Evolving Toward the Next Generation Internet: Challenges in the Path Forward”, IEEE Communications Magazine, pp 72-76, Vol. 40, No. 7, July 2002

## Section - c.1 Network Congestion and QoS Management Strategies (Stewart Personick, et. al.)

- [10] Personick, S.D., “**Combining Circuit and Packet Switching in Next Generation Internet Backbone Networks**” [invited talk], 2001 Topical Meeting on Photonics in Switching, Monterey CA, June 13-15, 2001. (see appendix page 82)
- [11] Hongyuan Shi and Harish Sethu, “On Scheduling Real-Time Traffic under Controlled Load Service in an Integrated Services Internet,” *Journal of Communications and Networks*, vol. 5, no. 1, pages 100-108 March 2003.

## Section - c.2 QoS Architectures and Mechanisms (Harish Sethu, et. al.)

- [12] Adam J. O'Donnell and Harish Sethu, “**Congestion Control, Differentiated Services, and Efficient Capacity Management through a Novel Pricing Strategy**,” to appear in *Computer Communications*. (see appendix page 87)
- [13] Hongyuan Shi and Harish Sethu, “An Evaluation of Timestamp-Based Packet Schedulers Using a Novel Measure of Instantaneous Fairness”, to appear in *Proceedings of the Workshop on End-to-end Service Differentiation*, held in conjunction with the *IEEE International Performance, Computing, and Communications Conference (IPCCC)*, April 2003, Phoenix, Arizona, USA.
- [14] Yunkai Zhou and Harish Sethu, “Toward End-to-End Fairness: A Framework for the Allocation of Multiple Prioritized Resources,” to appear in *Proceedings of the Workshop on End-to-end Service Differentiation*, held in conjunction with the *IEEE International Performance, Computing, and Communications Conference (IPCCC)*, April 2003, Phoenix, Arizona, USA.
- [15] Salil S. Kanhere, Harish Sethu and Alpa B. Parekh “**Fair, Efficient and Packet Scheduling using Elastic Round Robin**,” in *IEEE Transactions on Parallel and Distributed Systems* March 2002, volume 13, number 3, pages 324-336. (see appendix page 114)
- [16] Salil S. Kanhere and Harish Sethu, “**Low-Latency Guaranteed-Rate Scheduling using Elastic Round Robin**,” in *Computer Communications* September 2002, volume 25, number 14, pages 1315-1322. (see appendix page 127)
- [17] Yunkai Zhou and Harish Sethu, “On the Relationship between Absolute and Relative Fairness Bounds,” in *IEEE Communication Letters* January 2002, volume 6, number 1, pages 37-39.

- [18] Salil S. Kanhere and Harish Sethu, "On the Latency Bound of Pre-Order Deficit Round Robin," in *Proceedings of the IEEE International Conference on Local Computer Networks (LCN)* November 2002, Tampa, Florida, USA.
- [19] Salil S. Kanhere and Harish Sethu, "On the Latency Bound of Deficit Round Robin," in *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN)* October 2002, Miami, Florida, USA.
- [20] Yunkai Zhou, Madhusudan Hosaagrahara and Harish Sethu, "Opportunity-Based Deficit Round Robin: A Novel Packet Scheduling Strategy for Wireless Networks," in *Proceedings of the Workshop on High-Performance Switching and Routing (HPSR)* May 2002, Kobe, Japan.
- [21] Adam J. O'Donnell and Harish Sethu, "A Novel, Practical Pricing Strategy for Congestion Control and Differentiated Services," in *Proceedings of the IEEE International Conference on Communications (ICC)* April 2002, New York City, New York, USA.
- [22] Yunkai Zhou and Harish Sethu, "On the Effectiveness of Buffer Sharing in Multimedia Server Network Switches with Self-Similar Traffic," in *Proceedings of the IEEE International Conference on Communications (ICC)* April 2002, New York City, New York, USA.
- [23] Hongyuan Shi and Harish Sethu, "**On Scheduling Real-Time Traffic under Controlled Load Service in an Integrated Services Internet,**" in *Proceedings of the IEEE Workshop on High-Performance Switching and Routing (HPSR)* May 2001, Dallas, Texas, USA. (see appendix page 135)
- [24] Salil S. Kanhere and Harish Sethu, "Fair, Efficient and Low-Latency Packet Scheduling using Nested Deficit Round Robin," in *Proceedings of the IEEE Workshop on High-Performance Switching and Routing (HPSR)* May 2001, Dallas, Texas, USA.
- [25] Salil S. Kanhere and Harish Sethu, "Fair, Efficient and Scalable Scheduling without Per-Flow State," in *Proceedings of the IEEE International Performance, Computing and Communications Conference (IPCCC)* April 2001, Phoenix, Arizona, USA.
- [26] Madhusudan Hosaagrahara and Harish Sethu, "A Novel Enhancement to the Virtual Clock Algorithm and its Analysis," in *Proceedings of the Applied Telecommunications Symposium (part of Advanced Simulation Technologies Conference)* April 2001, Seattle, Washington, USA.
- [27] Madhusudan Hosaagrahara and Harish Sethu, "A Simulation Study of the Impact of VC Merging on the Delay in an MPLS Domain," in *Proceedings of the Applied Telecommunications Symposium (part of Advanced Simulation Technologies Conference)* April 2001, Seattle, Washington, USA.
- [28] Hongyuan Shi and Harish Sethu, "An Evaluation of the Longest-Queue-First Scheduler for Routers in a Differentiated Services Domain," in *Proceedings of the Applied*

*Telecommunications Symposium (part of Advanced Simulation Technologies Conference)*  
April 2001, Seattle, Washington, USA.

- [29] Hongyuan Shi and Harish Sethu, "Virtual Circuit Blocking Probabilities in an ATM Banyan Network with bxb Switching Elements," in *Proceedings of the Applied Telecommunications Symposium (part of Advanced Simulation Technologies Conference)* April 2001, Seattle, Washington, USA.
- [30] Harish Sethu, Hongyuan Shi, Salil S. Kanhere and Alpa B. Parekh, "A Round-Robin Scheduling Strategy for Reduced Delays in Wormhole Switches with Virtual Lanes," in *Proceedings of the International Conference on Communications in Computing* June 2000, Las Vegas, Nevada, USA.
- [31] Yunkai Zhou and Harish Sethu, "**A Simulation Study of the Impact of Switching Systems on Self-Similar Properties of Traffic**," in *Proceedings of the IEEE Workshop on Statistical Signal and Array Processing (SSAP)* August 2000, Pocono Manor, Pennsylvania, USA. (see appendix page 140)
- [32] Yunkai Zhou and Harish Sethu, "Performance of Shared Output Queueing in ATM Switches under Self-Similar Traffic," in *Proceedings of the Applied Telecommunications Symposium (part of Advanced Simulation Technologies Conference)* April 2000, Washington, D.C., USA.
- [33] Yunkai Zhou and Harish Sethu, "Trade-Offs Between Delay Jitter and System Design Properties of Switching Networks," in *Proceedings of the Applied Telecommunications Symposium (part of Advanced Simulation Technologies Conference)* April 2000, Washington, D.C., USA.

### **Section - c.3 Embedded Network Processing (Jonathan Smith, et. al.)**

- [34] Steve J. Muir, **Piglet: An Operating System for Network Appliances**, Ph.D. Thesis  
University of Pennsylvania, 2001. (see appendix page 145)

### **Section - c.4 IP over WDM (Mohamed Ali, et. al.)**

- [35] C. Assi, A. Shami, R. Kurtz, and M. A. Ali, "**Optical networking and real-time provisioning: An integrated vision for the next-generation Internet**," *IEEE Network Magazine*, July/Aug. 2001, Vol. 15 No. 4, PP. 36-45. (see appendix page 262)
- [36] C. Assi, Y. Ye, A. Shami, S. Dixit, I. Habib, and M. A. Ali, "Designing a survivable IP-over-WDM network," *OptiComm 2001*, PP. 1-11, Denver, Colorado, Aug. 01.
- [37] A. Shami, Y. Ye, C. Assi, S. Dixit, and M. A. Ali, "A simple distributed signaling algorithm for real-time provisioning of optical channels in IP-centric DWDM-based optical networks," *OptiComm 2001*, PP. 112-119, Denver, Colorado, Aug. 01.

- [38] Yinghua Ye, C. Assi, S. Dixit, and M. A. Ali, "A Simple Dynamic Integrated Provisioning/Protection Scheme in IP over WDM Networks," *IEEE Communication Magazine*, Dec. 2001.
- [39] A. Shami, C. Assi, and M. A. Ali, "Dynamic wavelength provisioning in DWDM-based optical networks," The 5<sup>th</sup> Working-Conference on Optical Network Design and Modeling "ONDM 2001", Vienna, Austria.
- [40] A. Shami, Yinghua Ye, C. Assi, and M. A. Ali, "Multi-Path Based Distributed Routing Algorithm for WDM Routed Networks," ECOC 2001, Amsterdam, The Netherlands.
- [41] C. Assi, Y. Ye, A. Shami, S. Dixit, and M. A. Ali, "On the Merit of IP/MPLS Protection/Restoration in IP over WDM networks" *IEEE GLOBECOM'01*, San Antonio, Texas.
- [42] A. Shami, C. Assi, I. Habib, and M. A. Ali, "On the merits of Flooding/Parallel probing-based signaling algorithms for fast automatic setup and tear-down of paths in IP/MPLS-over-optical-networks," *IEEE GLOBECOM'01*, San Antonio, Texas.
- [43] C. Assi, A. Shami, I. Habib, "and M. A. Ali, "GMPLS constraint-based routing of low rate traffic streams at the IP/MPLS layer in future IP-centric optical networks," to be presented at ICC 2002, NY, NY.
- [44] A. Shami, C. Assi, I. Habib, and M. A. Ali, "Performance Evaluation of Two GMPLS-Based Distributed Control and Management Protocols for Dynamic Lightpath Provisioning in Future IP Networks," to be presented at ICC 2002, NY, NY.
- [45] M. A. Ali, C. Assi, A. Shami, and R. Kurtz, "Architectural Options for Next-Generation Networking Paradigm: Is Optical Internet the answer?", Special Issue IP-over-WDM, Photonic Network Communications Journal, Nov, 2000, PP. 7-21.
- [46] Yinghua Ye, S. Dixit, and M. A. Ali, "On Joint Protection/Restoration in a hybrid, IP-centric DWDM-based data optical Network," *IEEE Communication Magazine*, June 2000, PP. 174-183.

#### **Section - c.6 Wavelength agility and optical networks (Bahram Nabet, Janet Jackel, et. al.)**

- [47] Z. Zhang, C. Assi, A. Shami, and M. A. Ali, "Impact of wavelength converters on the performance of optical networks", Opticom 2000, Dallas, Texas, October 2000
- [48] X. Chen, B. Nabet, et. al. "**Resonant-cavity-enhanced heterostructure metal-semiconductor-metal photodetector**", *Applied Physics Letters*, Vol 80, Number 17, pages 3222-3224 April 29, 2002. (see appendix page 272)

- [49] P. Toliver, R. Runser, J. Young, and J. Jackel, **Experimental Field Trial of Waveband Switching and Transmission in a Transparent Reconfigurable Optical Network**, 2003 Optical Fiber Conference (OFC 2003), Atlanta, GA. (see appendix page 275)

## **Section - d.2 Cellular Observatory (J. Yasha Kresh, Banu Onaral, et. al)**

- [50] C. Gustafson, O. Tretiak, L. Bertrand and J. Nissanov **Design and Implementation of Software for Assembly and Browsing of 3D Brain Atlases** [Under review for publication]. (see appendix page 281)
- [51] Kresh, J.Y., Izrailtyan, I., Wechsler, A.S. "The Heart as a Complex Adaptive System" Second International Conference on Complex Systems, to be published in "Unifying Themes in Complex Systems", ed. Y. Bar-Yam, Perseus Books, 2001.
- [52] Onaral, B., *Kresh, J.Y.*, Luzuria, E. "Complex Biological Systems" Plenary Lecture: IEEE-EMBS Asia-Pacific Conference on Biomedical Engineering, September 26-28, 2000, China
- [53] Izrailtyan I., *Kresh J.Y.*, Brozena SC, Morris R.J., Wechsler A.S. "Early Detection of Acute Allograft Rejection by Linear and Nonlinear Analysis of Heart Rate Variability. Journal of Thoracic Cardiovascular Surgery 120:737-45, 2000
- [54] Rabbany, S.Y., *Kresh, J.Y.*, Noordergraaf, A. "Myocardial Wall Stress: Evaluation and Management." Cardiovascular Engineering (Journal for Extracorporeal Circulation, Assist Devices, Transplantation and Artificial Organs) Vol. 5(1), p. 3-10, 2000.
- [55] Kerkhof, P.L., *Kresh, J.Y.*, Li, J.K. "Partial Ventriculectomy, Myocardial Oxygen Consumption and Ejection Fraction: Cardiodynamic Considerations." European Heart Journal Vol 2 (Abstr Supp.), p. 422, 2000
- [56] S. Rangappa, J. Entwistle, A. Wechsler, J. Yasha Kresh, **Cardiomyocyte-mediated Contact Programs Human Mesenchymal Stem Cells to Express Cardiogenic Phenotype**; Oral presentation at the American Heart Association's Scientific Sessions, Chicago, Illinois, November 17-20, 2002. (see appendix page 294)
- [57] V. Noronha, C. Yarman, J. Yasha Kresh, B. Onaral, **Remote Monitoring of Cellular Network Assembly and Function**, (to be published) (see appendix page 322)

## **Section - e.1 All-Optical Networking Devices and Subsystems (Paul Prucnal, et. al.)**

- [58] B. C. Wang, V. Baby, W. Tong, L. Xu, M. Friedman, R. J. Runser, I. Glesk, and P. R. Prucnal, "A novel fast optical switch based on two cascaded Terahertz Optical Asymmetric Demultiplexers (TOAD)," Optics Express 10, 15 (2002). (see appendix page 326)

- [59] L. Xu, M. Yao, B. C. Wang, I. Glesk, and P. R. Prucnal, **"All-optical Clock Division with Mode-locked Figure-eight Laser Based on the Slow Carrier Recovery Rate in Semiconductor Optical Amplifier,"** IEEE Photonics Technology Letters 14, (3) 402-404 (2002). (see appendix page 335)
- [60] B. C. Wang, L. Xu, V. Baby, D. Zhou, R. J. Runser, I. Glesk, and P. R. Prucnal, **"Experimental Study on the Regeneration Capability of the Terahertz Optical Asymmetric Demultiplexer,"** Optics Communications 199, 83-88 (2001). (see appendix page 338)
- [61] I. Glesk, J. Runser, and P. R. Prucnal, **"New generation of devices for all-optical communications,"** Acta Physica Slovaca 51, (2) 151-162 (2001). (see appendix page 344)
- [62] R. J. Runser, D. Zhou, B. C. Wang, C. Coldwell, P. Toliver, K.-Li Deng, I. Glesk, and P. R. Prucnal, **"Interferometric Ultrafast SOA-based Optical Switches: From Devices to Applications,"** *Optical and Quantum Electronics* 33, No 7/10 841-874 (2001). Special Issue on Components for Ultrafast Communications. Invited paper. (see appendix page 356)
- [63] P. Toliver, I. Glesk, and P. R. Prucnal, **"All-optical clock and data separation technique for asynchronous packet-switched OTDM networks,"** Optical Communications 173, (1-6) 101-106 (2000). (see appendix page 390)
- [64] P. Toliver, R. J. Runser, I. Glesk, and P. R. Prucnal, **"Comparison of three nonlinear interferometric optical switch geometries,"** Optics Communications 175, (4-6) 365-373 (2000). (see appendix page 396)
- [65] D. Zhou, I. Glesk, and P. R. Prucnal, "Optical Impulse Response of Semiconductor Optical Amplifiers in a Counter-Propagation Mach-Zehnder Switch," *OSA Trends in Optics and Photonics Vol. 32, Photonics in Switching*, Paul R. Prucnal and Daniel J. Blumenthal, eds. (Optical Society of America, Washington DC, 2000), pp. 144-151.
- [66] I. Glesk, "TOAD an All-Optical Switch for Ultrashort Pulse Signal Processing and Networks," CLEO/PR-2001 Makahuri Messe, Chiba, Japan, June 15-19, 2001.
- [67] I. Glesk, R. Runser, and P. R. Prucnal, "New Trends in Optical Communications", in *12th Czech-Slovak-Polish Optical Conference on Wave and Quantum Aspects of Contemporary Optics*, Jan Perina, Miroslav Hrabovský, Jaromír Krepelka, Editors, Proceedings of SPIE Vol. 4356, pp. 102-110, (2000).

## **APPENDIX**

### **Published Papers**



# DEMONSTRATION OF A 1 Tb/s OPTICAL PACKET SWITCH FABRIC (80 \* 12.5 Gb/s), SCALABLE TO 128 Tb/s (6400 \* 20 Gb/s)

Jürgen Gripp, Pietro Bernasconi, Calvin Chan, Karl L. Sherman, and Martin Zirngibl

Bell-Laboratories, Lucent Technologies,  
791 Holmdel-Keyport Rd, Holmdel, NJ  
(mz@lucent.com)

*Abstract: We demonstrated a 1 Tb/s optical packet switch fabric consisting of an 80x80 array waveguide grating and fast tunable lasers, modulated at 12.5 Gb/s. The architecture is scalable to 128 Tb/s throughput.*

To scale cross-connects beyond one Tb/s throughput, the industry has now embraced optical fabrics /1/ based on either MEMS or micro bubble technology. These fabrics feature switching times on the order of milliseconds, which is insufficient for packet switching. For multi-Tb/s packet switches, an optical switch fabric with nanosecond setup time is needed.

A number of electro-optical switch fabrics have recently been proposed /2,3/. However, these fabrics require active switching in the core. Alternatively, all-optical switching has also been studied, together with the unavoidable network redesign /4,5/. In this paper, we present an optical packet switch fabric with potential multi-terabit throughput, which uses a passive optical core and maintains compatibility with existing networks. In addition, it is based on commercially available components.

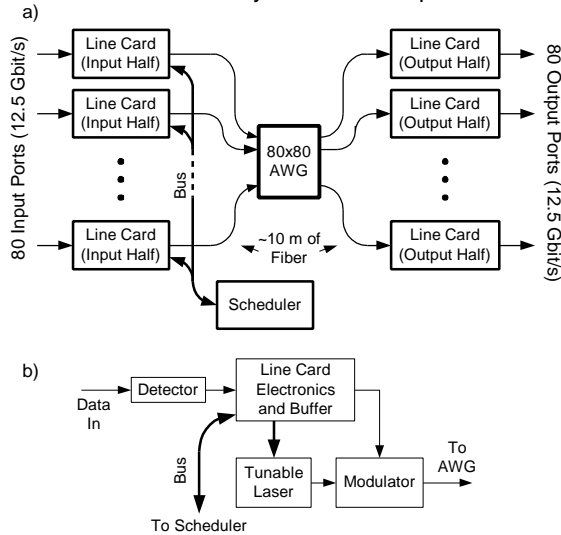


Figure 1. a) Switch architecture. b) Input half of a line card.

The switching in this fabric (fig.1), is achieved using the well-known wavelength routing properties of array waveguide gratings (AWGs) /6/. A tunable laser followed by an external modulator is connected to each input port of an NxN AWG. By tuning the wavelength of the laser, the data is wavelength-routed to one and only one output port. The result is a strictly non-blocking NxN crossbar switch with a switching time equal to the longest tuning time of all the lasers. Such a fabric has several intrinsic advantages: The core of the switch is completely passive, thus eliminating the issue of power consumption and reducing the likelihood of fabric failure. The initial cost of such a switch is low and port-units can be added when they are needed. All the active electronics, except the

scheduler, reside on the port cards, which can be far away from the core, thus solving the power density and real estate problems of conventional electrical switches. The fabric is bit-rate transparent up to the optical channel bandwidth.

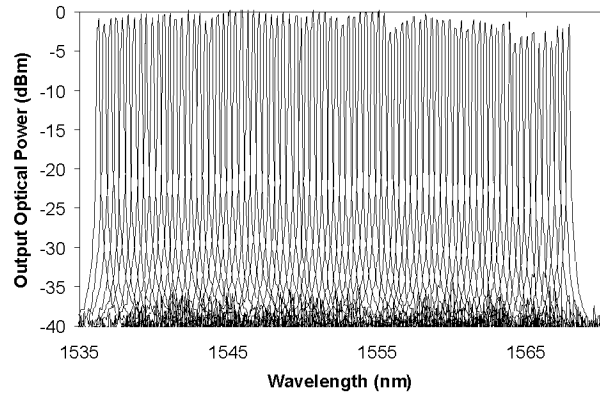


Figure 2. Tunable laser spectrum.

The intrinsic throughput of the current architecture is given by the number of ports times the modulation data rate. In this paper, we demonstrate an 80 port switch with 12.5 Gb/s data rate or a throughput of 1 Tb/s. However, 20 Gb/s can be supported by the actual optical channel bandwidth of 25 GHz and even higher data rates can be achieved by a proper reshaping of the optical AWG passband. Finally, this architecture can be scaled by a factor of N by using the full optical connectivity of an NxN AWG /7/. In this case, each input line card of fig. 1 is replaced by N line cards, whose outputs are power combined, and each output line card by a power splitter, N fast tunable filters, and N output line cards. For our current 80x80 AWG, this extended architecture reaches a maximum throughput of  $80^2 * 20 \text{ Gb/s} = 128 \text{ Tb/s}$ .

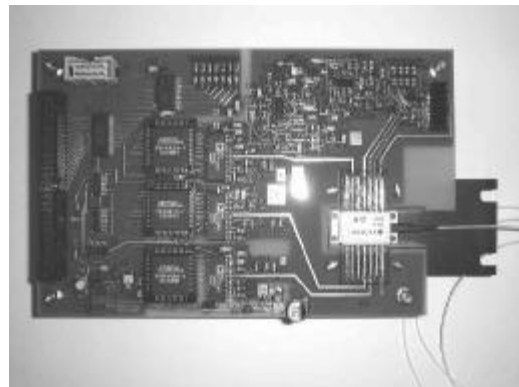


Figure 3. Photo of a tunable laser board.

The AWG was manufactured using a standard design platform for SiO<sub>2</sub> waveguide technology [8] with a 50 GHz channel spacing, a crosstalk level of -35 dB and a worst-case insertion loss of 7 dB. The fast wavelength switching is achieved with commercially available multi-section DBR lasers. The lasers can tune over 44 nm and reach optical power levels close to 0 dBm (fig. 2). We developed high-speed driver boards (fig. 3) that allow tuning with nanosecond rise times (fig. 4). To enable external digital control, the boards contain programmable logic devices that store the current settings for all wavelength channels in lookup tables. We have not noticed any wavelength drift of the devices over the course of our experiments. Should wavelength drift become an issue, periodic updating of the current tables could be implemented.

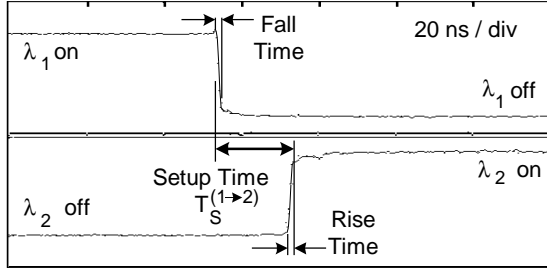


Figure 4. Rise, fall, and setup time for a sample transition.

To demonstrate the essential features of the optical part of the switch fabric, we proceeded in three steps: First, we switched a laser between different wavelengths and modulated packets on each wavelength with an external modulator. Second, we sent data packets through the AWG and received them without errors and without losing clock. Third, we repeated the same test while populating more ports of the AWG with signals to evaluate the impact of crosstalk.

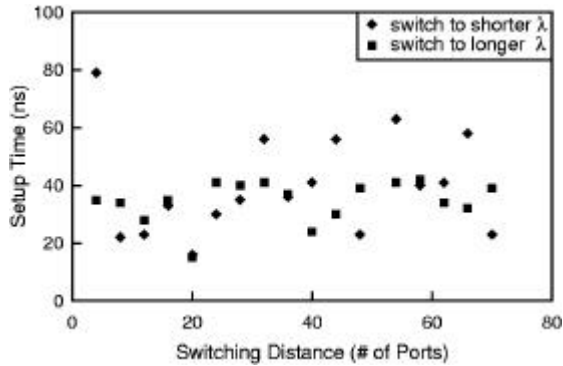


Figure 5. Setup times for for error-free ( $<10^{-11}$ ) packet switching.

Digital pulse generators supplied control and gating signals to a tunable laser board and a BER test set. The laser switched between two wavelengths  $\lambda_i$  and  $\lambda_j$  once every 2  $\mu$ s (fig. 4), and its light was modulated and sent into two ports of the AWG. The two ports were chosen such that light at  $\lambda_i$  or  $\lambda_j$  reached the same output port from either one or the other inputs, and then continued to the BER receiver. The BER pattern generator was then used to send pseudo random bit sequences with a data rate of either 10 or 12.5 Gb/s and a word length of  $2^{31}-1$  through the AWG. By moving the edges of the gating signal with respect to the laser trigger, we measured the maximum windows of error-free data transmission, both for data arriving on  $\lambda_i$  and on  $\lambda_j$ . Changing the data rate had no effect on the position of the edges. The lengths of the remaining gaps corresponded to the setup times  $T_s^{(i \rightarrow j)}$

and  $T_s^{(j \rightarrow i)}$  of the laser, for the transitions from  $\lambda_i$  to  $\lambda_j$  and from  $\lambda_j$  to  $\lambda_i$  respectively (fig. 4).

We measured a random selection of switching transitions with different wavelength separations between  $\lambda_i$  and  $\lambda_j$ . The setup times of these transitions ranged from 15 to 80 ns (fig. 5). The global setup time  $T_s$  for a fully loaded fabric has to be only slightly larger than the largest individual setup time  $T_s^{(i \rightarrow j)}$ . Although only a full characterization of  $80 \times 79 = 6320$  transitions can determine this number with certainty, the measured subset indicated that a setup time of 100 ns would be sufficient. This would add 5% overhead to the switch fabric for a switching period of 2  $\mu$ s. The switching period is compatible with high-capacity scheduling algorithms that aggregate data packets to avoid scheduler bottlenecks [9].

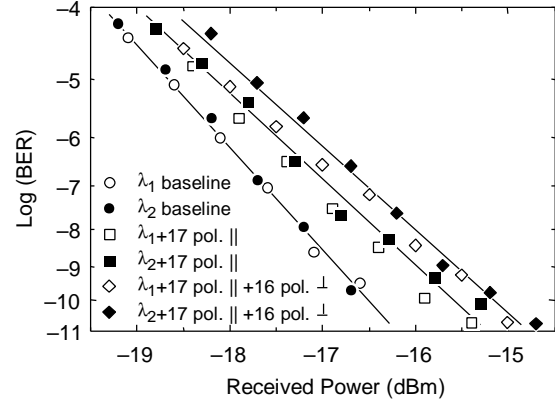


Figure 6. Bit-error-rate curves for the AWG.

We examined crosstalk by adding up to 32 signals to open ports of the AWG in different combinations and by varying the received optical power (fig. 6). Launching 16 copolarized and 16 orthogonally polarized channels of the same wavelength into other ports of the AWG resulted in less than 1.5 dB power penalty due to coherent crosstalk. Note that the probability of having 16 equal wavelength channels is less than  $1.7 \times 10^{-14}$  for random traffic. It could be eliminated through a simple modification of the scheduling algorithm. Incoherent crosstalk is not believed to be a problem with the AWG out of band rejection of -35 dB, even if the fabric is fully loaded with 80 channels. The measured results show that this optical switch fabric is well suited to provide the basis for the next generation of multi-terabit packet switches.

## References

- /1/ W. v. Parys, P. Arijis, and P. Demeester, OFC'00, ThO5-2, p217
- /2/ Y. Maeno et al., ECOC'99, p118
- /3/ N. Yamanaka et al., IEICE Trans. Commun., vol. E83B, p1488, 2000
- /4/ X. Jiang X. P. Chen, and A. E. Willner, IEEE Photon. Technol. Lett., vol. 10, p1638, 1998
- /5/ P. B. Hansen, S. L. Danielsen, and K. E. Stubkjaer, ECOC'98, p591
- /6/ M. Smit, Int J. Optoelectron., vol. 12, p25, 1998
- /7/ C. Dragone, IEEE Photon. Technol. Lett., vol. 3, p812, 1991
- /8/ P. Bernasconi et al., IEEE J. Lightwave Technol., vol.18, p985, 2000
- /9/ K. Kar et al., HOTI 8, presentation 1.3, 2000

This project is supported in part by DARPA / NGI grant no. F30602-00-2-0501.

# Demonstration of a 1.2 Tb/s Optical Packet Switch Fabric (32 \* 40 Gb/s) based on 40 Gb/s Burst-Mode Clock-Data-Recovery, Fast Tunable Lasers, and a high-performance NxN AWG

J. Gripp (1), M. Duelk (1), J. Simsarian (1), S. Chandrasekhar (1), P. Bernasconi (1),  
A. Bhardwaj (1), Y. Su (1), K. Sherman (1), L. Buhl (1), E. Laskowski (1),  
M. Cappuzzo (1), L. Stulz (1), M. Zirngibl (1), O. Laznicka (2),  
T. Link (3), R. Seitz (3), P. Mayer (3), and M. Berger (3)

(1) Bell-Laboratories, Lucent Technologies, 791 Holmdel-Keyport Rd, Holmdel, NJ, USA ([mz@lucent.com](mailto:mz@lucent.com))

(2) Internetworking Systems, Lucent Technologies, 55 Fairbanks Boulevard, Marlborough, MA, USA

(3) Bell-Laboratories, Lucent Technologies, Thurn & Taxis Strasse 10, Nuremberg, Germany

*Abstract: We demonstrate a 1.2 Tb/s optical packet switch fabric based on burst-mode clock-data-recovery at 40 Gb/s with packet separations of up to 400 ns and lock times under 5 ns, fast wavelength switching between 32 channels in less than 46 ns, and a 42x42 AWG with a worst-case loss of 4.2 dB.*

Optical switch fabrics are becoming increasingly popular for cross-connects with more than 1 Tb/s throughput because of the limited scalability of electrical backplanes. Packet switches have traditionally lagged cross-connects in total throughput by one order of magnitude, but they will also need optical fabrics to achieve multiple Tb/s capacity [1,2]. Whereas cross-connects work with millisecond switching times, packet switches require nanoseconds. The introduction of 40 Gb/s services in the network will intensify the push towards optical fabrics.

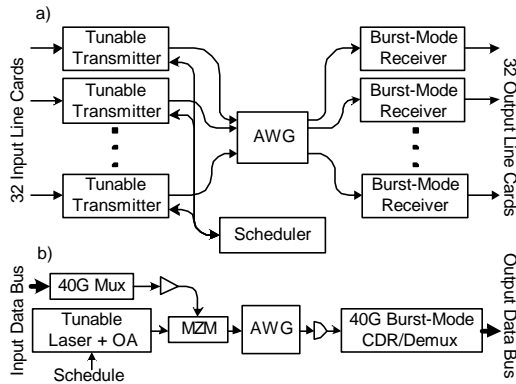
Recently, we demonstrated a multi-terabit optical packet switch fabric based on commercially available components with 10 Gb/s line speed [3]. Here, we report on significant improvements of this technology, notably fast phase and clock recovery at 40 Gb/s, improved tuning speed of the laser and a low-loss, low-crosstalk AWG. Together, these results clearly demonstrate the feasibility of a fast, 32 x 40 Gb/s strictly non-blocking switch fabric.

through the AWG to their individual output ports. At each output port a receiver with burst-mode clock-data-recovery (CDR) and an electrical Demux converts the optical packet back into an electrical one and sends it to the output line card.

The switching time of this fabric is determined by the longest tuning time of the lasers. Since no data can be transmitted during tuning, the receivers have to tolerate this dead time between data packets. In addition, the phase of the incoming bits changes from packet to packet, thus requiring fast-phase-recovery on the receiver side.

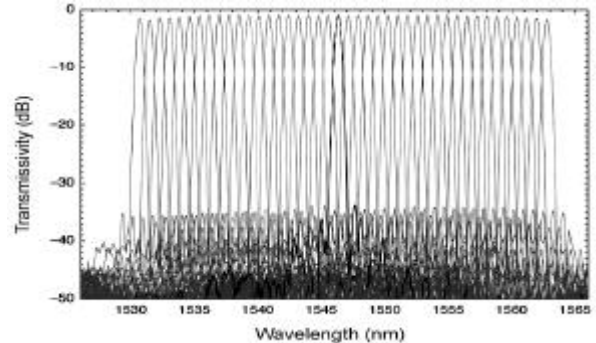
Following, we first describe the low-loss, low-crosstalk AWG and the receiver performance that are important for the power budget of the system. Then we show results on fast wavelength tuning and burst-mode reception of 40 Gb/s data packets that withstands dead times well above the tuning times of the lasers.

The passive 42x42 AWG has been fabricated using standard silicon bench technology. The channels are spaced by 100 GHz and the transmission passbands are Gaussian with a FWHM of 51 GHz. The AWG has no periodic frequency response. Among NxN AWGs, our device shows unprecedented low-loss and low-crosstalk [5,6]. The worst-case total insertion loss of the packaged device is less than 4.2 dB, while PDL is less than 0.23 dB. The polarization dependent frequency shift is less than 1.5 GHz. The total crosstalk level has been measured to be less than -29 dB at the center of the signal passband. This value does not change within the whole passband once the contributions due to the adjacent channels are suppressed. This suppression is easily achieved via polarization multiplexing [3]. Figure 2 shows a spectrum of the device.



**Figure 1. a) Switch architecture. b) One data path. OA: optical amplifier. MZM: Mach-Zehnder modulator.**

Figure 1a shows the architecture of the fabric, which is equivalent to a strictly non-blocking NxN crossbar. The switching relies on the well-known wavelength routing properties of array waveguide gratings (AWGs) [4]. M input line cards send data to tunable transmitters that in turn connect to the input ports of an NxN AWG ( $N \geq M$ ). Burst-mode receivers on the output ports of the AWG send the data to M output line cards. Figure 1b depicts one optical data path. Incoming data packets modulate the amplified light of a tunable laser. The packets then travel into one input port of the AWG. By tuning the wavelength, the packets are routed



**Figure 2. Spectrum of the 42x42 port AWG. The worst port-to-port loss is less than 4.2 dB (fully packaged).**

The sensitivity of the receiver unit is determined by the conversion gain of 30 V/W of the photo receiver and the single-ended input sensitivity of the CDR/Demux of 25 mV. This combination results in a back-to-back sensitivity of  $-4.7$  dBm at a BER of  $10^{-9}$ , as shown in fig. 3. The measurements were taken with a pseudo random bit sequence (PRBS) with a word length of  $2^{31}-1$  bits. The spectral filtering caused by the AWG passband results in 1 dB penalty. Per-channel optical amplification is currently needed to overcome all the losses between the tunable laser and the receiver. We anticipate however, that improvements in the output power of the tunable laser and better sensitivities of the photo receiver will eventually eliminate the need for optical amplification.

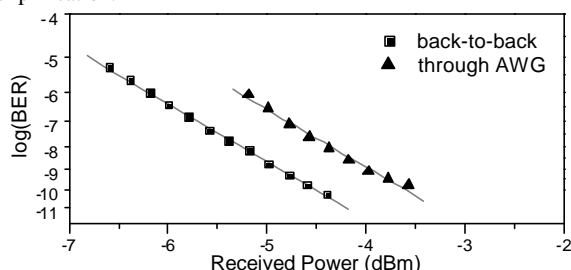


Figure 3. Receiver penalty due to the switch fabric.

The lasers are commercially available multi-section DBR lasers with a tuning range of 40 nm and output power close to 0 dBm. We have improved the high-speed driver boards described in [3] by adding high-speed drivers at the outputs of the D/A converters. We calibrated 32 channels with a spacing of 100 GHz and measured the time it takes the lasers to switch from each channel to every other channel. Figure 4 shows the resulting switching times of one laser for all possible  $32 \times 31 = 992$  combinations. The laser switches in less than 46 ns, with the majority of the channels switching between 10 and 30 ns. To our knowledge, this is the smallest reported worst-case switching time for this number of channels [3,7].

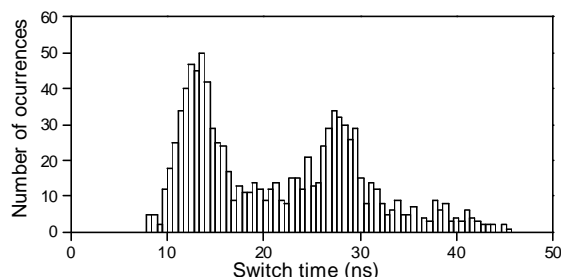


Figure 4 Switching time histogram for a laser with 32 calibrated channels with 100 GHz spacing. All channel combinations switched in less than 46 ns.

The 40 Gb/s CDR/Demux is a single SiGe chip, preceded by a limiting amplifier [8]. The CDR consists of a fast digital phase detector and a PLL based on an internal VCO. The Demux demultiplexes 40 Gb/s data into four 10 Gb/s tributaries. During the dead time between data packets, phase and frequency of the VCO drift and the PLL has to relock at the beginning of the next data packet. Since the Demux will produce errors, as long as the PLL is not fully locked, the data packets have to start with a training pattern that is longer than the lock time of the CDR. Figure 5 shows the structure of one time slot of the data stream that we used to test the CDR performance.

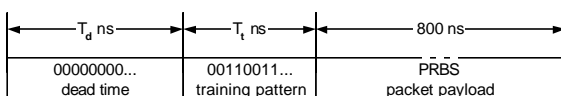


Figure 5. Structure of one time slot used in the CDR test.

The time slot is constantly repeated and consists of three sections, first a dead time, then a training pattern and then a payload. We varied the length of the dead time  $T_d$  between 50 ns and 100  $\mu$ s, and the length of the training pattern  $T_t$  between 25 ns and 200 ns. The payload was an 800 ns long PRBS pattern with a word length of  $2^{31}-1$  bits. The short word length was a result of limitations in the programmability of the pattern generator. However, the CDR/Demux is able to receive regular PRBS streams with a word length of at least  $2^{31}-1$  bits without errors, as shown in fig. 3.

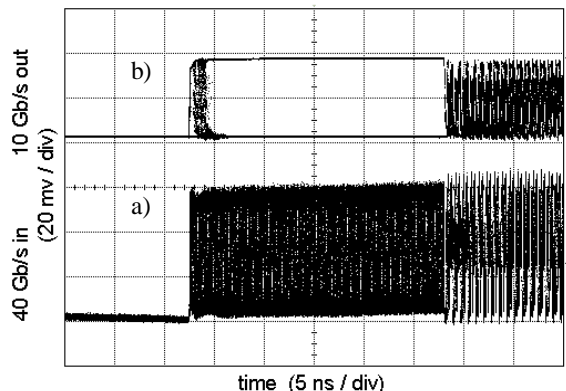


Figure 4. CDR/Demux behavior for  $T_d=50$  ns,  $T_t=25$  ns. a) Beginning of optical 40G data packets coming out of the fabric. b) One 10G output of the Demux, showing how the CDR locks to a tributary within 5 ns.

We tested the CDR/Demux with a subset of the setup shown in fig. 1, consisting of two transmitters that alternate in sending packets to one receiver. Trace a in fig. 4 shows the beginning of one data packet with the end of a 50 ns dead time (first 2.5 divisions), a 25 ns training pattern and the beginning of the PRBS payload. Trace b shows the behavior of one output of the CDR/Demux. During the dead time the Demux generates zeros. During the training pattern, the Demux output generates either a string of zeros or a string of ones, depending on the tributary to which the CDR locks. As long as the CDR is not locked, the output will show sudden transitions between ones and zeros, visible during the first 5 ns of the training pattern. The absence of transitions after the first 5 ns shows that the CDR has locked to one of the tributaries. We saw that up to 400 ns dead time the CDR was able to lock well within the 25 ns training period, even when varying the phase difference between packets continuously between 0 and  $2\pi$ . Since the lasers can switch in less than 46 ns, we have a wide design margin for the timing of the data packets in the optical fabric. The data format is compatible with timing requirements of current router architectures [9].

The results presented in this paper show that wavelength switching based optical packet fabrics with a line rate of 40 Gb/s and multi-Tb/s throughput are clearly feasible for the next generation of core packet switches.

## References

- /1/ N. McKeown, OFC' 01, MN1-1
- /2/ C. Guillemot et al., ECOC' 98, p83
- /3/ J. Gripp et al., ECOC' 00, postdeadline paper 2.7
- /4/ C. Dragone, IEEE Photon. Technol. Lett., vol. 3, p812, 1991
- /5/ K. Okamoto et al., Electron. Lett., vol.33, p1865, 1997
- /6/ P. Bernasconi et al., J. Lightwave Technol., vol.18, p985, 2000
- /7/ O. Lavrova et al., ECOC' 00, p23
- /8/ M. Reinhold et al., ISSCC 2001 paper 5.6
- /9/ K. Kar et al., HOTI 8, presentation 1.3, 2000

This project is supported in part by DARPA / NGI grant no. F30602-00-2-0501.

# 4 x 4 DEMONSTRATION OF A 1.2 Tb/s (32 x 40 Gb/s) OPTICAL SWITCH FABRIC FOR MULTI-Tb/s PACKET ROUTERS

J. Gripp (1), M. Duelk (1), J. Simsarian (1), P. Bernasconi (1), A. Bhardwaj (1), K. Sherman (1), K. Dreyer (1),  
M. Zirngibl (1) and O. Laznicka (2)

(1) Lucent Technologies, Bell-Laboratories, 791 Holmdel-Keyport Rd, Holmdel, NJ, USA (mz@lucent.com)

(2) Lucent Technologies, Internetworking Systems, 1 Robbins Rd, Westford, MA, USA

**Abstract** We demonstrate error-free, asynchronous packet transmission and fast simultaneous switching on four 40 Gb/s ports in a 32-port optical switch fabric. The sensitivity under switching is better than -3 dBm for all four burst-mode receivers.

## Introduction

Optical switch fabrics will enable next-generation multi-protocol switches and packet routers to reach multi-Tb/s throughputs. As routers move from single-shelf designs to multiple shelves or racks, optical interconnects become unavoidable [1]. Optical fabrics based on array waveguide gratings and fast wavelength tuning offer the advantage of replacing large active fabric cores [2] with an inexpensive, passive optical device. This reduces space and power requirements and permits a pay-as-you-grow architecture [3].

We have previously reported asynchronous 40 Gb/s packet transmission through such a fabric [4]. In this paper we demonstrate simultaneous operation of four 40 Gb/s “line cards” (packet transmitters and burst-mode receivers) in a fabric capable of 1.2 Tb/s throughput (32 line cards x 40 Gb/s).

## Setup

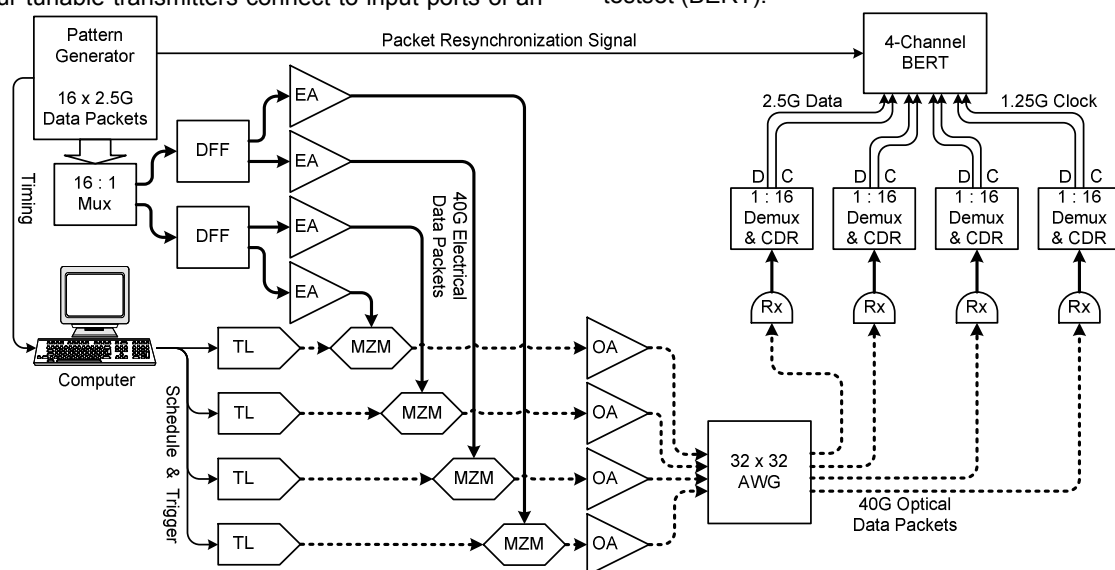
Figure 1 shows the setup of the optical fabric. Even though we only equip four ports, all components of the fabric are designed to support 32 ports. The basic architecture has been described previously [4]. Four tunable transmitters connect to input ports of an

array waveguide grating (AWG). The four output ports of the AWG connect to burst-mode receivers.

Each transmitter consists of a fast tunable laser module (TL), a LiNbO<sub>3</sub> Mach-Zehnder modulator (MZM), and an Erbium doped fiber amplifier (OA). The lasers are capable of tuning to 32 different wavelength channels spaced by 100 GHz in less than 50 ns [5]. Each transmitter receives electrical 40 Gb/s data packets from a 16 x 2.5 Gb/s pattern generator followed by a 16:1 multiplexer. To generate four electrical data streams, we duplicate the differential outputs of the Mux with delay flip-flops (DFFs). By setting an appropriate modulator bias, we obtain non-inverted data packets on all four transmitters.

The AWG has a channel spacing of 100 GHz and Gaussian passbands with a FWHM of 50 GHz. The worst-case fiber-to-fiber insertion loss is less than 4.2 dB [6].

Each burst-mode receiver consists of a photoreceiver and a 1:16 demultiplexer with fast clock-data recovery (CDR). We connect one 2.5 Gb/s data output from each Demux together with a recovered 1.25 GHz clock to one channel of a 4-channel bit-error rate testset (BERT).



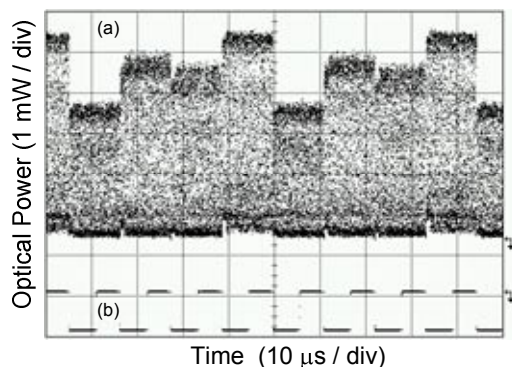
**Figure 1:** Experimental setup. TL: tunable laser; AWG: array waveguide grating; DFF: delay flip-flop; EA: electrical amplifier; OA: optical amplifier; MZM: Mach-Zehnder modulator; Rx: photo receiver; CDR: clock-data recovery; BERT: bit-error-rate testset.



The pattern generator sends timing signals to a computer that in turn provides the tunable lasers with a switching schedule and trigger signals. We chose a simple round-robin schedule to send data packets from each transmitter to each receiver every 4 switching cycles. In addition, the pattern generator sends a signal to the BERT to resynchronize to the payload of every new packet.

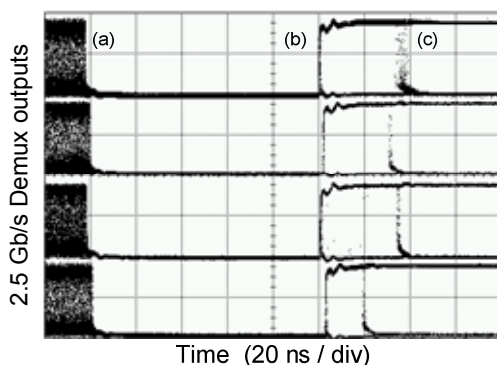
### Operation and Results

Figure 2 (a) shows the optical data packets, at one output port of the AWG. Wavelength-dependent power of the lasers and path-dependent loss through the AWG cause power variations from packet to packet. Trace (b) shows that the BERT synchronizes successfully on all data packets (transition to high).



**Figure 2:** (a): Optical data packets after AWG. (b): Synchronization output signal of the BERT.

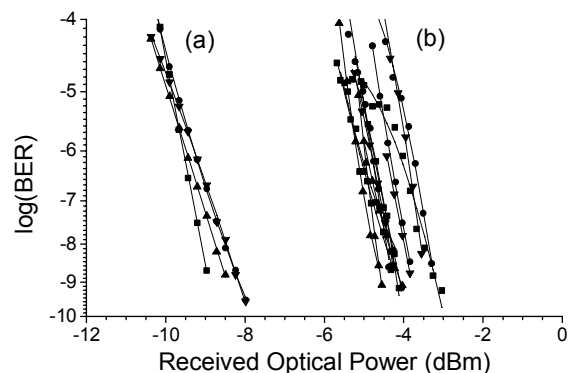
For a PRBS pattern with  $2^7-1$  pattern length, the BERT requires approximately 6000 bits for synchronization. At a sampling rate of 1.25 Gb/s this translates to roughly 5 μs. We therefore chose a payload length of 10 μs so that bit-errors are measured during 50% of the packet. We furthermore program the pattern generator to precede the payload with a 1 μs training pattern (00110011...) and to follow it with a 100 ns dead-time during which all lasers switch their wavelengths.



**Figure 3:** One 2.5 Gb/s data output for each one of the 1:16 Demultiplexers. (a): start of dead-time. (b): end of dead-time. (c): CDR locks to new data packet.

Figure 3 shows the electrical 2.5 Gb/s data outputs of the 1:16 demultiplexers under burst-mode packet reception. At point (a) the previous packet turns off. At point (b), the new data packet arrives, starting with the training pattern. The 00110011 sequence is down-sampled either to constant 0 or constant 1. After point (c), the CDR circuit has adjusted to the phase of the new packet. The time for locking onto the new packet is between 20 and 40 ns.

Figure 4 shows BER measurements comparing (a) the back-to-back receiver sensitivity with (b) the sensitivity under switching. For (a) we connect a transmitter directly with each receiver and transmit continuous PRBS data. For (b) we switch data packets and measure the sensitivity for all 16 possible combinations of four receivers at four output ports. The received optical power is time averaged over the power variations from packet to packet. We observe a spread of 1.5 dB, between all receivers-port combinations and a total penalty of about 5 dB. This penalty is due to filtering in the AWG [4], power variations between data packets (fig. 2), and the fact that the optimum decision threshold varies from packet to packet. In all cases the sensitivity (for BER <  $10^{-9}$ ) under switching is better than -3 dBm.



**Figure 4:** BER curves. (a): synchronous back-to-back transmission. (b): asynchronous packet switching.

### Conclusions

We have demonstrated simultaneous packet switching between four 40 Gb/s line cards in an optical switch fabric capable of 1.2 Tb/s throughput. The asynchronous burst-mode packet reception has a sensitivity of better than -3 dBm for all combinations. These results present significant progress towards the demonstration of a fully-loaded terabit-class optical switch fabric.

### References

- 1 McKeown, OFC 2001, MN1-1
- 2 N. Sahri et al., OFC 2001, postdeadline paper 32
- 3 M. Duell et al., Proceedings SPIE 4872, 2002
- 4 J. Gripp et al., ECOC'01, postdeadline paper A.1.7
- 5 J. Simsarian et al., ECOC'02, paper 3.3.6
- 6 P. Bernasconi et al, JLT, vol. 18, no. 7, p985, 2000

# A WIDELY TUNABLE LASER TRANSMITTER WITH FAST, ACCURATE SWITCHING BETWEEN ALL CHANNEL COMBINATIONS

J. E. Simsarian (1), A. Bhardwaj (1), K. Dreyer (1), J. Gripp (1), O. Laznicka (2), K. Sherman (1), Y. Su (1), C. Webb (1), L. Zhang (1), and M. Zirngibl (1)

1 : Lucent Technologies, Bell Laboratories, 791 Holmdel-Keyport Rd., Holmdel, NJ, USA, jesses@lucent.com

2 : Internetworking Systems, Lucent Technologies, 1 Robbins Road, Westford, MA, USA

**Abstract** We demonstrate widely tunable laser transmitters that can access 32 ITU channels with 100 GHz spacing and accurately switch between all channel combinations in less than 50 ns. The compact modules use commercially available components. We also show the correlation of switching times to laser tuning currents.

## Introduction

Fast tunable lasers, with nanosecond switching speeds enable optical switch fabrics for multi-terabit/s packet routers [1]. These switch fabrics use tunable lasers to route data packets based on wavelength switching. A critical efficiency requirement is that the switching time must be short compared to the packet length. Furthermore, the laser must switch between any channel combination within a specified time.

There has been substantial prior work on tunable lasers that demonstrated fast switching between some channels [2-6]. However, data was not reported for all of the possible switching combinations. Also, the frequency accuracy of the laser under switching conditions (a critical parameter for deployment in an optical system) was not reported in addition to the static accuracy.

Previously, we reported switching times of 50 ns for all  $32 \times 31 = 992$  possible combinations of a 32 channel system [1]. In this paper, we report results on two different lasers, investigate the impact of the driver circuit on the switching, and show the correlation of the switching times and tuning currents.

## Laser Module

We use a commercially available vertical Grating assisted codirectional Coupler with rear Sampled grating Reflector (GCSR) semiconductor laser, with a tuning range of up to 40 nm [7]. The laser has four separate sections: gain, coupler, reflector, and phase. The gain section produces light emission, whereas the coupler, reflector, and phase sections allow for wavelength tuning. Due to manufacturing variations, each laser must be calibrated to determine the tuning currents for the ITU wavelengths.

We have designed and fabricated a compact printed circuit board that controls the laser and enables the fast wavelength switching. All of the components are commercially available, and the butterfly-packaged laser is directly mounted on the board. Figure 1 shows a block diagram of the control circuit. A Field Programmable Gate Array (FPGA) stores the lookup

table containing the laser calibration. To set the laser to a particular channel, the channel number is sent as parallel data on the control bus to the FPGA. The lookup table converts the channel number to three tuning currents that undergo 10-bit digital to analog (D/A) conversion. The analog currents are amplified and sent to the tuning sections of the laser.

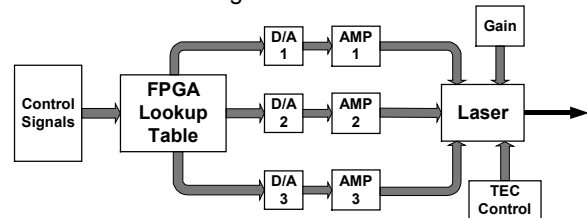


Figure 1 Block diagram of the module electronics

Here we report the switching results from two lasers. We generated the lookup table for laser #1 using a calibration system and software that we developed. We calibrated the static transmission wavelengths to within  $\pm 1$  GHz of 32 ITU channels from 1528.77 nm to 1553.33 nm. For laser #2, we used the calibration table of tuning currents generated by the laser manufacturer, which we transferred to the module.

## Switching Results

We define the switching time as the interval between when the wavelength leaves the source channel and arrives at the destination channel to within a specified accuracy. We measure the switching time for every channel combination automatically using a tunable filter and photodetector. The transmission percentage through the filter gives the frequency accuracy of the laser.

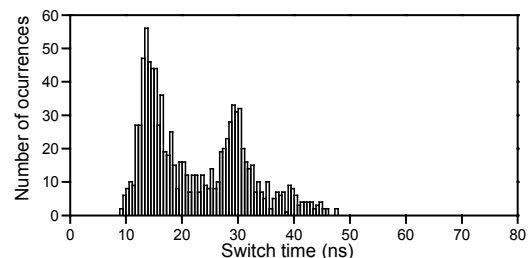
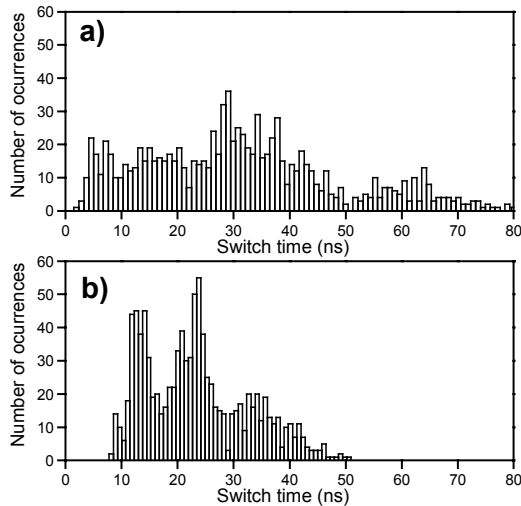


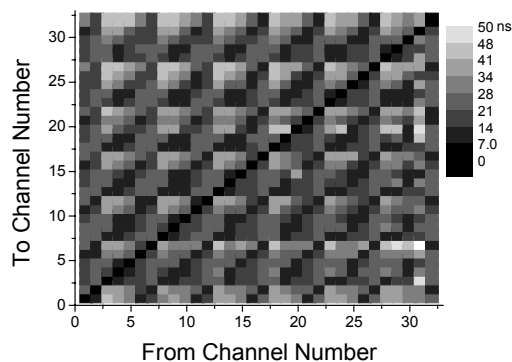
Figure 2 Histogram of the switching times for laser #1, using a high output-impedance current source for all three tuning currents

A histogram of switching times for all 992 channel combinations of laser #1 is plotted in Fig. 2. The three amplifier sections of the circuit (see Fig. 1) are high output-impedance current sources. The accuracy for switching to the destination channel is  $\pm 10$  GHz. All switching times are below 50 ns.



**Figure 3** Histogram of the switching times for laser #2 when **a)** all three amplifier circuits are high-output impedance current sources **b)** the amplifier circuit for the reflector section is a low-impedance source with  $R_{out} = 20 \Omega$

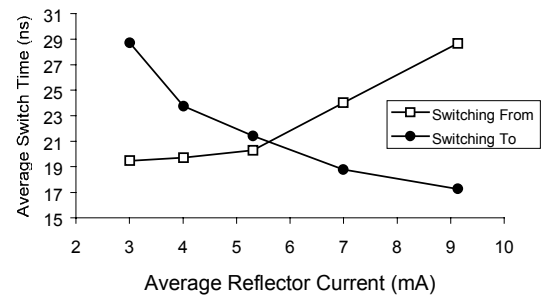
When we measure the switching times for laser #2 using the three high output-impedance amplifier circuits, we find longer switching times with a maximum of 80 ns (Fig. 3a). This is due to the low reflector currents for this particular laser. The high dynamic resistance of the tuning section at low current as well as the chip and parasitic capacitances cause long settling times when switching from high to low currents. By changing the reflector section driver to a low output-impedance amplifier circuit with  $R_{out} = 20 \Omega$ , the maximum switching time is reduced to 51 ns (Fig. 3b). The switching accuracy to the destination channel is  $\pm 12$  GHz for this measurement.



**Figure 4.** The switching times (represented by the gray scale) for all 992 channel combinations of laser #2

We investigate the performance-limiting factors by plotting the switching-time matrix for all channel combinations in Fig. 4. The data set shown is the same as for the histogram of Fig. 3b. A periodicity is evident in the matrix as seen in the diagonal, horizontal, and vertical lines of lower switching times and isolated areas of higher switching times. The periodicity mostly follows that of the reflector current.

The correlation of the switching times to the reflector current can be extracted by further data processing. We first calculate the average switching times to every channel (row averages of the matrix) and from every channel (column averages of the matrix). Then we group the channels with similar reflector currents and calculate the average switching time and reflector current for each group. The results are plotted in Fig. 5. It is evident that the longer switching times occur when going from high to low reflector currents. This indicates that the settling of the reflector current is limiting the switching time. Optimization of the amplifier circuit for the reflector may result in further improvement.



**Figure 5** Plot showing the correlation between the switching time and the reflector current

## Summary

We demonstrate a high-volume manufacturable, fast-tunable laser transmitter that switches between any channel combination in less than 50 ns. The reflector driver circuit currently limits the switching time. While the steady-state accuracy of the transmitter is better than  $\pm 1$  GHz, under wavelength switching conditions, the worst-case accuracy can be  $\pm 10$  GHz.

## References

- 1 J. Gripp et al., ECOC'01, post-deadline ThA4.8
- 2 C.-K. Chan et al., IEEE Photon. Technol. Lett., vol. 13, (2001), p. 729
- 3 O. Lavrova et al., ECOC'00, vol. 2, p. 169
- 4 Y. Fukashiro et al., OFC'00, WM43-1, p. 338
- 5 R. O'Dowd et al., IEEE J. Selected Topics in Quantum Electron., vol. 7, (2001), p. 259
- 6 P. D. Biernacki et al., J. Lightwave Technol., vol. 17, (1999), p. 1222
- 7 P.-J. Rigole et al. IEEE Electron. Lett., vol. 32, (1996), p. 2352

This project is supported in part by DARPA / NGI grant no. F30602-00-2-0501.



# LONG-TERM WAVELENGTH SWITCHING MEASUREMENTS WITH RANDOM SCHEDULES ON FAST TUNABLE LASERS

A. Bhardwaj, J. Gripp, J. E. Simsarian and M. Zirngibl

Lucent Technologies, Bell Laboratories, 791 Holmdel-Keyport Road, Holmdel, NJ 07733

ashishb@lucent.com

**Abstract** We demonstrate 26 hours of error-free wavelength switching in less than 50 ns between 32 ITU channels with 100 GHz spacing using a GCSR laser. This corresponds to a packet-loss rate of less than  $2 \times 10^{-11}$ . We also examine the effect of different switching schedules, an important issue for commercial switch fabrics.

## Introduction

Fast and widely tunable lasers will be an important component in next-generation switch fabrics. In combination with NxN array waveguide gratings and nanosecond wavelength switching, they enable packet routers with multi-terabit/s throughputs [1]. Other approaches include e.g. SOA-array based fabrics [2], but they rely on components that are not commercially available.

In previous work, we reported less than 50 ns switching speed on all possible combinations of a 32-channel system with  $\pm 10$  GHz accuracy as well as data transmission and switching of 1  $\mu$ s data packets carrying 40 Gb/s data [1]. Other groups have also reported fast switching [3], but not long-term switching measurements.

In this paper we demonstrate error-free continuous switching over time periods of up to 26 hours. Furthermore, we test the impact of different switching schedules on the laser performance. Reliable long-term operation under these varying conditions is essential for commercial deployment.

## Fast Tunable GCSR Laser

The laser used for these measurements is a commercially available vertical Grating assisted co-directional Coupler with Rear Sampled grating Reflector (GCSR) semiconductor laser [4]. It is digitally controlled through a circuit board consisting of a Field Programmable Gate Array (FPGA), fast digital-to-analog converters (D/As), and amplifiers [2]. The D/As provide currents to three tuning sections of the laser to control the wavelength. Before performing the switching measurements, we calibrate the laser and program the FPGA to hold lookup tables for 32 ITU channels, from 1528.77 nm (ch. 0) to 1553.33 nm (ch. 31), covering a range of 24.5 nm.

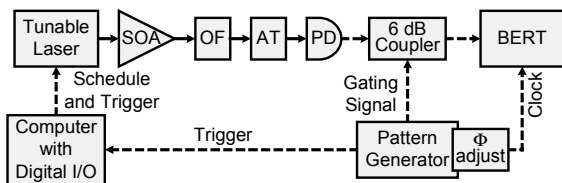


Figure 1: Setup. SOA: semiconductor optical amplifier, OF: optical bandpass filter, AT: attenuator, PD: photodetector, BERT: bit-error rate detector.

## Setup

Figure 1 shows the switching setup. A computer with a digital I/O card applies a periodic switching schedule together with a trigger to the tunable laser. The light from the laser is amplified using a Semiconductor Optical Amplifier (SOA) and then passes through a tunable optical bandpass filter with a full width half maximum of 0.2 nm. We arbitrarily select the ITU wavelength 1544.53 nm (ch. 20) as the center wavelength of the filter. The light transmitted by the filter is attenuated and photodetected with 1 GHz bandwidth. A pattern generator provides the overall timing for the experiment. It triggers the computer to change the laser wavelength once every 1  $\mu$ s, clocks the BERT at 40 MHz, and gates the output of the photodetector. The BERT then samples the sum of the photodetector and gating signals (see fig. 2 (c)). When the gating signal is high, the BERT measures a 1-bit, regardless of the detected optical power. If the laser doesn't switch within the allotted time, the BERT counts at least one error.

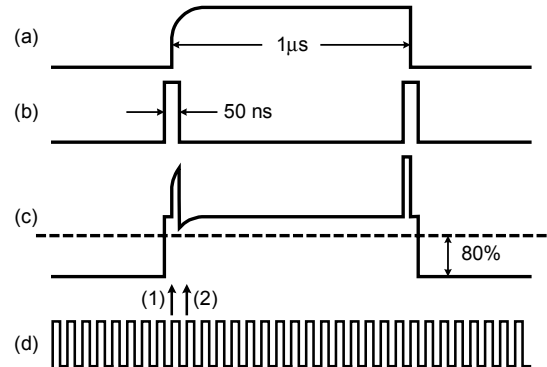


Figure 2: Timing diagram. (a) Signal from the photo detector (b) gating signal (c) Signal at BERT input, the dotted line shows the decision threshold (d) clock signal. Arrow (1): BERT measures a 1-bit regardless of jitter in the laser turn-on. Arrow (2): If the laser switches too slowly the BERT counts at least one error.

## Switching Schedules

We generate schedules that switch the laser continuously once every microsecond between channel 20 and one of the 31 other channels. In the following, we employ two different schedules. The

first is a linear schedule which cycles through the 31 other channels, resulting in the following pattern: 0,20,1,20,2,20...31,20,0,20... The second is a random schedule in which the other channels are randomly chosen for a long sequence of 20,000  $\mu$ s. With these schedules we can examine if the switching history impacts the switching behavior of the laser.

### Operation

Figure 2 shows the timing of the signals used in the switching measurements. The photodetector measures approximately a rectangular signal with 2  $\mu$ s periodicity, shown as trace (a). Every time the laser switches to channel 20, the signal goes from a low to a high value. The BERT is programmed to expect a string of N 0-bits followed by a string of N 1-bits, where N is the number of clock cycles per packet, 40 in our case. We set the threshold to 80% of the maximum laser power. This level is reached as soon as the laser wavelength approaches channel 20 to within  $\pm 8$  GHz. If the laser doesn't switch fast enough, the BERT will measure a 0-bit where a 1-bit is expected and count an error. The gating signal ensures that the BERT does not count errors during the switching time window. The phase of the clock signal determines how fast the laser has to switch to avoid an error count by the BERT. Since a slow laser transition will cause at least one error to be counted per 2  $\mu$ s, we can calculate an upper limit for the packet-loss rate (PLR) to be  $PLR \leq 2 * N * BER$ , where BER is the bit-error rate. As the name indicates, the packet-loss rate describes the probability of losing data packets due to slow laser transitions. We change the clock phase and record the PLR as a function of maximum switching time.

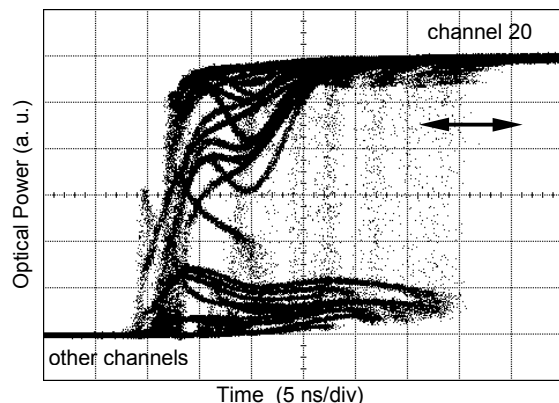


Figure 3: Random-schedule switch diagram. Discrete traces for transitions from the 31 other channels to channel 20 (1544.53 nm) are visible. The arrow shows the path on which the PLR curve of fig 4 is obtained.

### Results

Figure 3 shows a switch diagram obtained using the

random schedule. It shows how the laser switches to channel 20. Discrete traces are visible, indicating that the process of switching from the other channels to channel 20 is highly repeatable. The equivalent switch diagram for the linear schedule looks identical to figure 3.

Figure 4 shows packet-loss rate curves for the linear and random schedules. We varied the clock phase and thus adjusted the maximum allowed switching time between 35 and 50 ns. By doing so, we effectively sampled fig. 3 along the arrow. Above 46 ns, the laser generated no errors. The choice of schedule caused a penalty of well under 1 ns. We ran the measurement for 26 hours with a switching time window of 50 ns and counted no errors. This corresponds to a PLR below  $2 \cdot 10^{-11}$ .

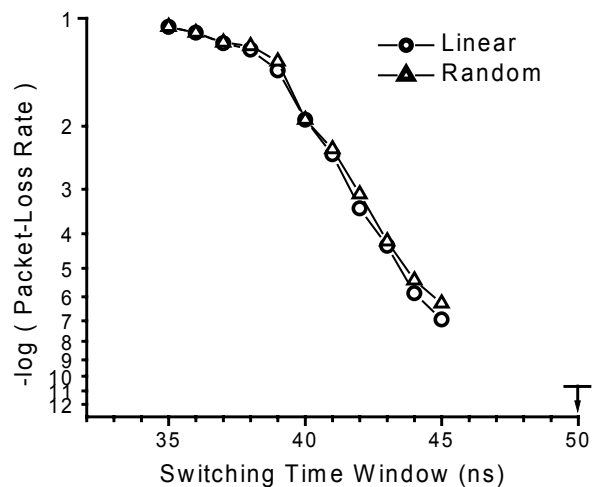


Figure 4: Packet-Loss Rate curves. The arrow corresponds to 26 hours of error-free operation.

### Conclusion

We have demonstrated long-term switching of a fast tunable laser. We presented a packet-loss rate curve for switching to one particular ITU channel from the 31 other channels. It shows that a time window of 50 ns provides ample margin for error-free wavelength switching. Full laser characterization requires 32 such curves, one for each destination channel. This will be performed in the near future. However, the results clearly indicate that fast tunable lasers are a reliable, enabling technology for next-generation switch fabrics.

### References

- 1 J. Gripp et al., ECOC'01, ThA4.8
- 2 D. Chiaroni et al., ECOC'01, ThA4.11
- 3 O. Lavrova et al., ECOC'00, vol. 2, p. 169
- 4 P.-J. Rigole et al., IEEE Electron. Lett., vol. 32, (1996), p. 2352

This project is supported in part by DARPA / NGI grant no. F30602-00-2-0501.

# 16-CHANNEL DIGITALLY TUNABLE PACKET SWITCHING TRANSMITTER WITH SUB-NANOSECOND SWITCHING TIME

M. Kauer (1), M. Girault (2), J. Leuthold (1), J. Honthaaas (2), O. Pellegrini (2), C. Goullancourt (2), and M. Zirngibl (1)

1 : Bell Laboratories, Lucent Technologies, 791 Holmdel-Keyport Road, Holmdel, NJ 07733, USA,  
mkauer@lucent.com

2 : NetTest, 45 avenue Jean Jaures, BP 81, 78344 Les Clayes sous Bois, France, marc.girault@nettest.com

**Abstract** We present a 16-channel 100 GHz-spacing digitally tunable packet switching transmitter based on an external-cavity laser. The transmitter can switch within 0.8 ns between its channels at 0 dBm output power and <0.7 dB penalty.

## Introduction

Tunable lasers are considered important components for next generation DWDM networks. In particular, they enable fast optical switch fabrics for packet routers that route traffic on a packet-by-packet basis [1]. Grating Assisted Coupler and Reflector (GCSR) lasers that can be switched in less than 50 ns [1] and less than 5 ns [2] have been demonstrated recently. These lasers are tuned by current injection and hence it is difficult to maintain wavelength accuracy if one wants to reduce the switching time below a few nanoseconds. Here, we demonstrate a packet switching transmitter that can switch between its channels with 0.8 ns guard time and less than 0.7 dB penalty.

## External-cavity laser

The packet switching transmitter is based on a novel external-cavity laser design [3]. The external-cavity laser uses a 16-strip laser diode chip and a stationary diffraction grating (see Fig. 1). Its emission

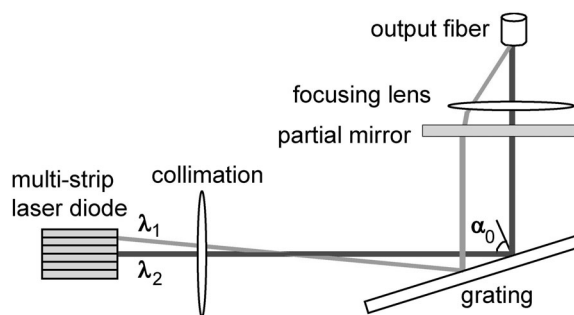


Figure 1: Schematic of the laser cavity.

wavelength depends on the position of the laser diode waveguide with respect to the grating, and thus can be switched by selecting the desired laser diode on the chip. The grating period, laser diode separation and average incidence angle  $\alpha_0$  are chosen such that the channels are on the 100 GHz ITU-grid (see Fig. 2, the currents for 0 dBm output power are between 45 and 79 mA). This laser has the characteristic

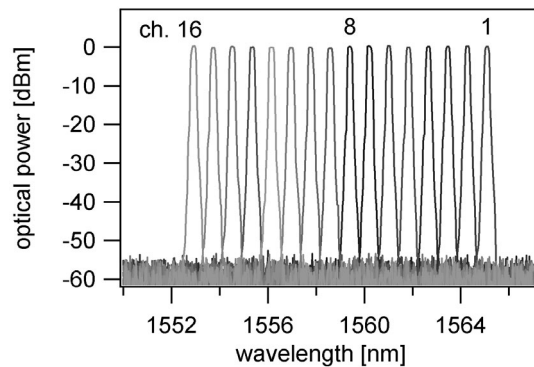


Figure 2: Superimposed spectra of all 16 channels with output power adjusted to 0 dBm.

advantages of external-cavity lasers, such as high output power (>7 dBm), high side-mode suppression ratio (>40 dB) and inherently narrow linewidth (<300 kHz) [3]. In addition to its excellent optical properties, the novel design offers three important benefits. Whereas conventional external-cavity lasers use rotating or translating gratings as the tuning mechanism, this laser does not contain any moving parts. This increases the long-term reliability of the laser and makes a wavelength locker unnecessary as the emission wavelengths are determined by the geometric properties of the cavity. Most importantly, the design of this laser enables very rapid channel switching, as there is no trade-off between tuning accuracy and switching speed.

## Experimental scheme

The experimental scheme consists of the transmitter, a demultiplexer and a receiver (see Fig. 3). We switch the laser between combinations of two wavelengths with a period of 3.2  $\mu$ s. The individual laser diode strips are driven by programmable pulse generators and DC-biased with a bias tee. We adjusted the switching current and the DC bias such as to obtain 0 dBm output power per channel during the on-period and an on/off-ratio larger than 12 dB. The DC-bias currents were just above threshold, between 33 and 38 mA. The laser output signal is externally

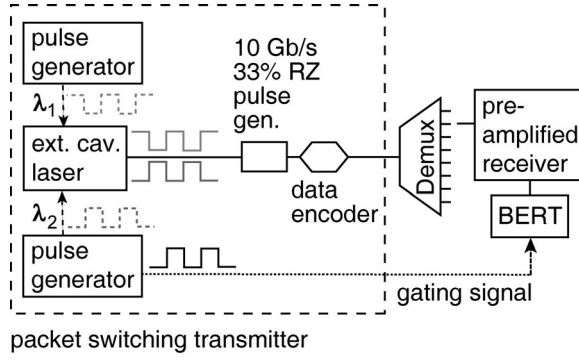


Figure 3: Experimental scheme.

modulated at 10 Gb/s by a programmed pseudo-random bit sequence (PRBS of  $2^{23}-1$ , limited by pattern generator memory). The PRBS is synchronized with the switching of the laser. After the laser we use a demultiplexer to select one wavelength for bit-error rate (BER) measurements. In order to measure the BER we apply a gating signal to the BER tester, thus taking into account only the data within the gating window. The gating window could be shifted with respect to the wavelength packets and also shortened to allow for an adjustable guard time between the two wavelengths. The timing jitter for the drive pulses and the gating window signal was less than 100 ps.

## Results and discussion

In order to determine the switching time we measure for each wavelength the maximum gating time with which error-free transmission can be achieved.

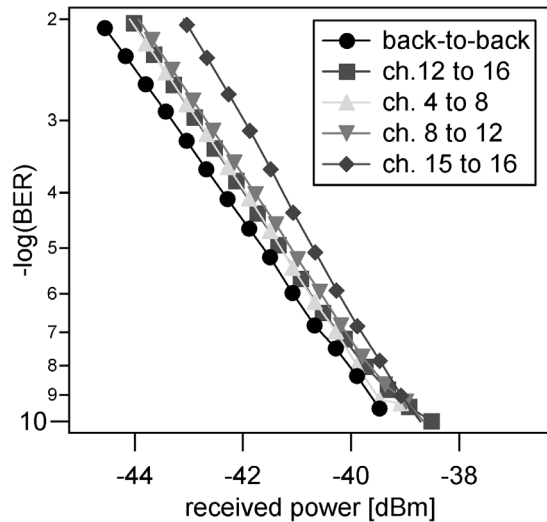


Figure 4: BER results for the switched signals with 0.8 ns guard time compared to the cw back-to-back measurement.

From the measured gating times we deduce the guard time and find that error-free transmission is possible with a guard time of 0.8 ns between the wavelength packets. Figure 4 shows the correspond-

ing BER curves for four wavelength pairs across the laser tuning range. The penalty for a BER of  $10^{-9}$  is found to be less than 0.7 dB. The bit sequences and the gating window during switching for one particular wavelength pair are shown in Fig. 5. During the transition, the signal within the gating window from the intended channel is at least 4.8 dB larger than the signal from the unwanted channel.

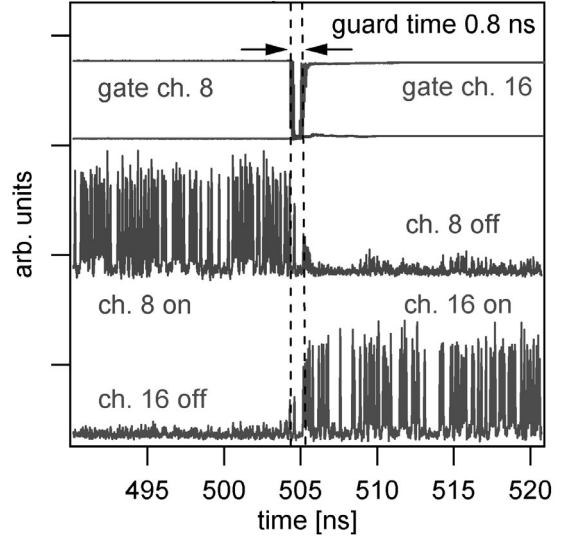


Figure 5: Bit sequence while switching from channel 8 to channel 16 and corresponding gating window.

We can achieve zero guard time by turning on a new channel while the old channel is still on, i.e. by overlapping the packets. We set the gating windows such that the guard time between the packets is zero and then adjust the laser drive pulses so as to achieve error-free operation. An overlap of 5 ns or less is sufficient to achieve zero guard time with a power penalty of less than 3 dB for a BER of  $10^{-9}$ . Cross-talk between the overlapped signals affects their quality and leads to the larger power penalty.

## Conclusions

In conclusion, we have demonstrated a digitally tunable packet switching transmitter based on a novel external-cavity laser design without moving parts. It can switch between wavelength packets with a guard time of 0.8 ns at 0 dBm output power and less than 0.7 dB penalty. This is significantly better than what is possible with other tunable lasers, including those based on current-injection.

We thank P. Fowler, C. Jones, J. Gripp, J. Simsarian, P. Bernasconi, A. Bhardwaj, M. Dülk, and K. Sherman.

## References

- 1 J. Gripp et al., ECOC 2001, ThA4.8 (post-deadline)
- 2 O. A. Lavrova et al., ECOC 2000, Tu 6.3.5
- 3 G. Souhaité et al., ECOC 2001, Tu.F.3.2

## **Combining Circuit and Packet Switching in Next Generation Internet Backbone Networks**

Stewart D. Personick  
Drexel University

This work is supported by AFRL/DARPA agreement No. F30602-00-2-0501

Traditional digital telephone networks utilize time-division-multiplex circuit switching to share physical network links among multiple telephone connections. In addition, circuit switching, in the form of re-arrangeable time-division-multiplexed paths, is commonly utilized to provide dedicated (but re-arrangeable via network management functionality) capacity between pairs of router ports in packet networks. When one employs byte-by-byte time division multiplexing (e.g., as in SONET), to create circuit switched paths between pairs of router ports, one can achieve the benefits of deterministic (as opposed to statistical) sharing of the capacity of the fibers or wavelengths that make up the physical interconnection fabric of a network. But this comes at the expense of deploying and operating both conventional byte-by-byte time division multiplexing equipment, and packet switching equipment; i.e., two, overlaid networks.

It is possible to combine statistical packet switching and circuit switching in the same switching mechanism. For example, the IEEE 1394 “Firewire” standard allows a portion of the capacity of a link to be allocated for circuit switched (pre-reserved) traffic, and the remaining capacity to be allocated for packet switched (unreserved) traffic. The Fiber Distributed Data Interface (FDDI) standard allows for the construction of a ring-shaped local area network in which a portion of the capacity of the ring is allocated for “isochronous” (circuit switched) traffic, and the remainder of the capacity of the ring is allocated to “asynchronous” (packet switched) traffic.

In this paper we describe a methodology for designing and implementing a wide area telecommunications network with a generalized physical topology... i.e., not limited to a ring or other type of typical local area network physical topology... that combines deterministic circuit switching of packets or bursts of telecommunications traffic with statistical packet or burst switching of telecommunications traffic, in a flexible manner, that achieves the guaranteed throughput and fixed delays associated with circuit switching, while still obtaining the statistical multiplexing and other benefits of packet switching. Since a common packet or burst format is used for both deterministic and statistical switching, one can avoid the need to deploy and operate two, overlaid physical infrastructures.

An illustrative example of this methodology is shown in Figure 1. In this example, there are what we are calling here (i.e., this is not standard terminology) “transitional” switching nodes and “core” switching nodes. Not shown in Figure 1 are: the traditional circuit and packet switched networks that route traffic to and from the transitional switching nodes, and thus use the network of Figure 1 as a backbone from transporting their traffic. The core switching nodes and the associated links provide connectivity between transitional switching nodes for carrying telecommunications traffic. The

telecommunications traffic traveling between transitional switching nodes consists of either fixed length (number of bytes) packets, or fixed length bursts of consecutive bytes that may include one or more variable length packets, plus filler bytes that fill up unneeded space in a burst. Variable-length packets or bursts that arrive at transitional nodes from other networks (not shown in Figure 1) can be repackaged, by the transitional nodes, into fixed length packets or bursts by using a combination of: segmentation, stuffing, and tunneling methods.

Fixed length packets or fixed length bursts being transported between transitional nodes are interleaved in time as shown in Figure 2. A cycle (equivalent to a “frame” in conventional byte-by-byte time division multiplexing) represents an interval of time that can accommodate a sequence of fixed length packets or a sequence of fixed length bursts. Each fixed length packet or fixed length burst occupies one of  $N$  time slots in the cycle. The cycle repeats at a fixed periodic rate:  $R$  (cycles per second); and traffic associated with a source-destination pair can be assigned some number of time slots per cycle. For illustration, we can assume that a cycle accommodates 64 fixed length packets or fixed length bursts in time slots numbered 0,1,2,...,63. However, the number of time slots in a cycle is a design parameter that can be selected, as desired, for specific network implementations. Traffic that is assigned two time slots out of the total number of time slots in a cycle would have two time slots per cycle (not just two time slots in one cycle) of capacity allocated to it for as long as those time slots are assigned to that traffic.

If a transitional switching node wishes to establish circuit switched connectivity to another transitional switching node, it signals (sends a circuit set-up request) to the core network using a signaling protocol. The network contains a network management capability. It can be implemented as a centralized network management system that manages the switching nodes and the assignment of time slots on core network links, or it can be implemented as a distributed network management system, with portions of its software-based functionality distributed among the switching nodes, or it can be implemented as a combination of these. The network management system of the network maintains network management data regarding the availability of time slots (i.e., time slots that have not been previously reserved) on the links of the core network. If a sufficient number of time slots are available on a sequence of links between the source transitional switching node and the destination transitional switching node, to provide the desired circuit capacity between those transitional switching nodes, then those time slots are reserved on those links for this new circuit switched connection. If the network management system cannot find a sufficient number of time slots on a set of links leading from the source to the destination, then the request to set up the new circuit switched connection is denied.

The circuits that are established between ports on transitional switching nodes can be used to carry packed switched traffic between those nodes, without any contention (i.e., no statistical multiplexing) in the core network that provides these circuits.

Each core switching node must manage the contention that occurs when two or more incoming links contain traffic for a given outgoing link. Even though there are enough

time slots per cycle on the outgoing link to accommodate all of the fixed length packets or fixed length bursts that have reserved capacity on that outgoing link, the incoming fixed length packets or fixed length bursts will generally need to be assigned to different time slot numbers on the outgoing link, than the time slot numbers they occupy on the incoming links. For example two fixed length packets on incoming links #1 and #5, respectively might both arrive in time slot 7 of their respective cycles. To direct both of these fixed length packets to output link #12, it will be necessary to move at least one of these incoming packets to a different time slot number, since they cannot both occupy the same time slot on the outgoing link. The subsystems that perform this rearrangement of time slots are called time slot interchangers (see Figure 3).

Using properly designed time slot interchangers, one can assign unused time slots in any links that are needed to establish new end-to-end connections, regardless of what existing connections are established.

The concept of assigning fixed length packets or fixed length bursts to time slots in a cycle is analogous to the process of assigning eight-bit bytes to reserved time slots in a “frame” in conventional time division switching and multiplexing. Much of the theory of conventional time division switching and multiplexing can be reused to create methodologies to perform such functions as setting up end-to-end connections (finding and reserving time slots) and time slot interchanging.

In addition to reserving time slots within the cycles of series of links to establish circuit switched end-to-end connections for transporting fixed length packets or fixed length bursts, one can also use time slots that have not been reserved within a cycle on a link, or that have been reserved, but which are not used to carry the associated circuit switched traffic during a single cycle, for packet-switched traffic (see Figure 4). Note that packet switched traffic, consisting of fixed length packets or fixed length bursts, must be buffered (temporarily stored) at each core switching node, while it awaits an unreserved or reserved, but not used time slot in a cycle of a suitable outgoing link. The destination address or some form of label is used to route fixed length packets or fixed length bursts that travel through the network as packet switched traffic using packet switching protocols.

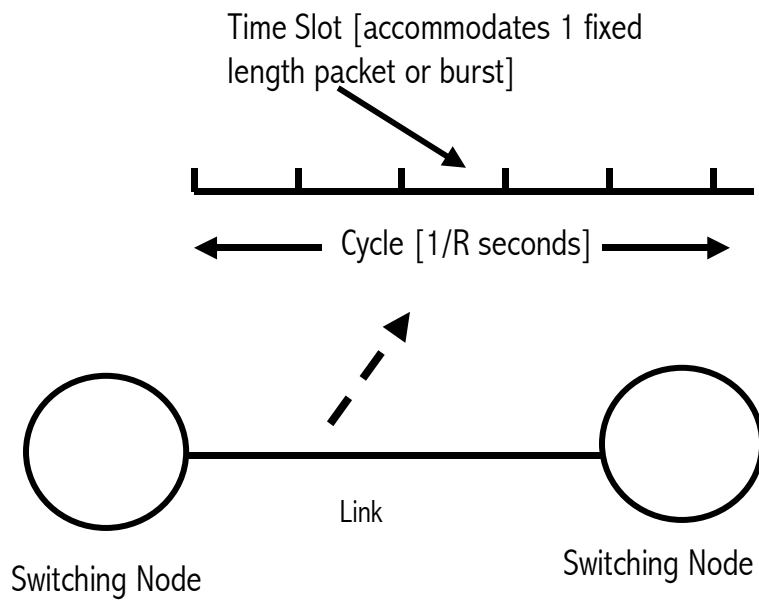
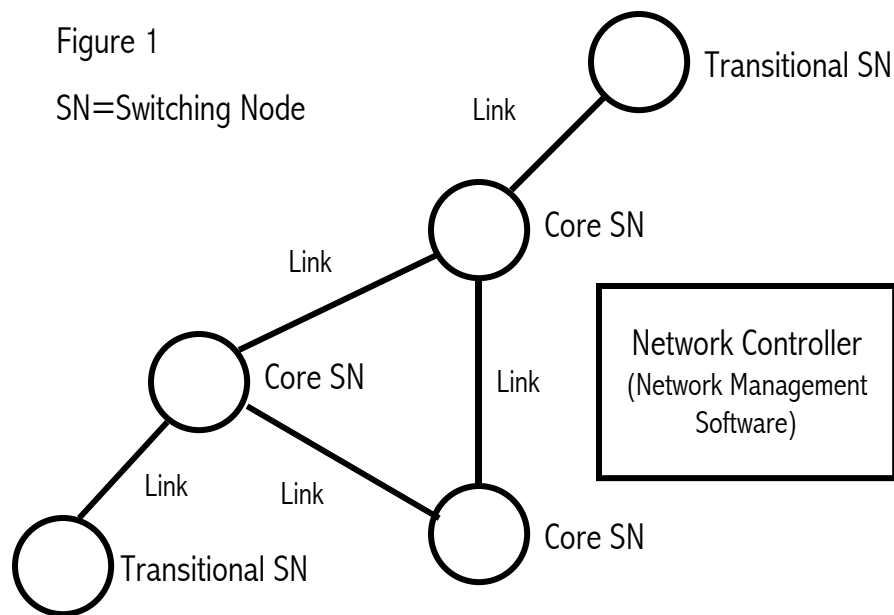


Figure 2



Figure 3 Time Slot Interchanger (TSI)

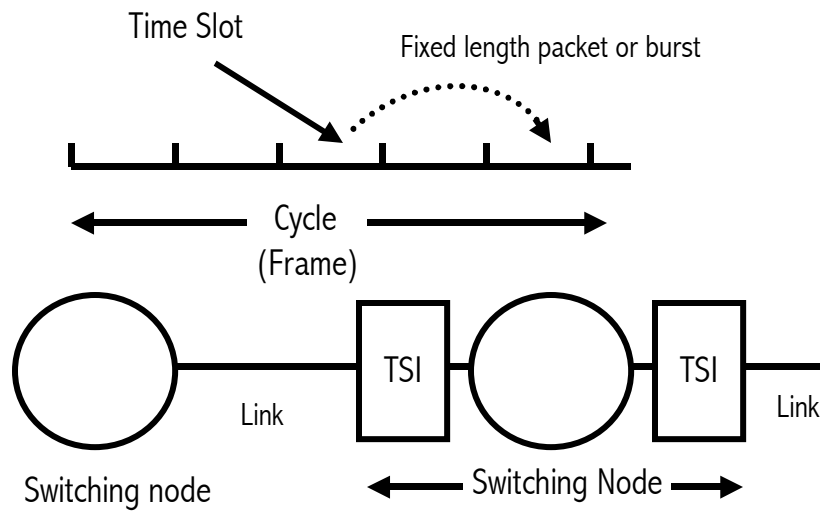
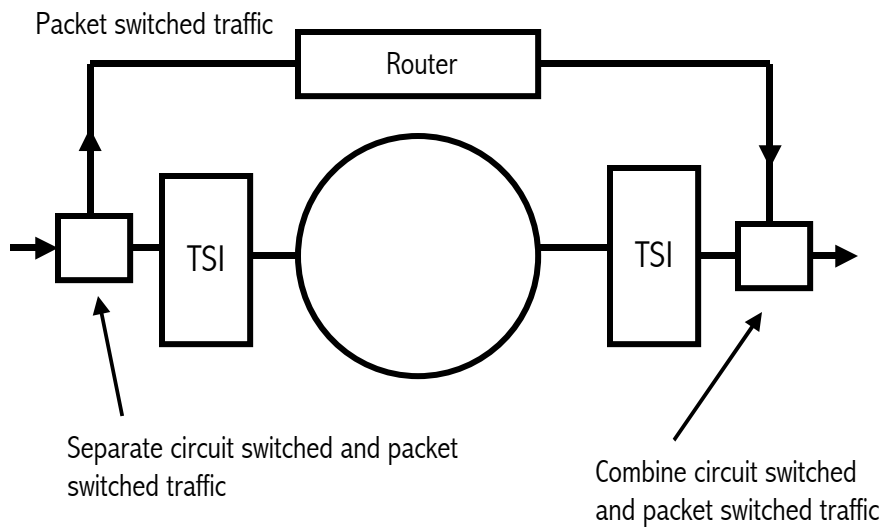


Figure 4 Combined Deterministic and Statistical Switching



# Congestion Control, Differentiated Services, and Efficient Capacity Management Through a Novel Pricing Strategy<sup>1</sup>

Adam J. O'Donnell, Harish Sethu<sup>\*</sup>

*Department of Electrical and Computer Engineering  
Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104.*

---

## Abstract

Pricing is an effective tool to control congestion and achieve QoS provisioning for multiple differentiated levels of service. In this paper, we propose a practical, flexible and computationally simple pricing strategy that can achieve QoS provisioning in Differentiated Services networks with multiple priority classes operating in an efficient economic market, while also maintaining stable transmission rates from end-users. In contrast to previous work, in which dynamic pricing strategies are based on the state of congestion alone, our strategy adds a separate price component for the preferential service received by a packet. This permits an efficient market for network resources and services, with the price charged being dependent upon both the cost of the resources and the dynamically changing demand for it. In addition, this automatically enforces efficient capacity management in the allocation of resources among the various service classes, leading to a user-centric approach where a user is not charged a higher price unless preferential service is *actually* delivered. Our analytical and simulation results demonstrate that, with the combination of user adaptation and our pricing strategy, differentiated services can be achieved with stable transmission rates. This paper concludes with a discussion of various operational issues associated with actual deployment of such a pricing strategy.

*Key words:* Internet economics, Differentiated Services (DiffServ), pricing, congestion control, Quality of Service (QoS), capacity management

---

---

<sup>\*</sup> Corresponding author.

*Email addresses:* adam@ece.drexel.edu (Adam J. O'Donnell),  
sethu@ece.drexel.edu (Harish Sethu).

<sup>1</sup> This work was supported in part by Drexel University's ECE Colehower Endowed Fellowship, United States National Science Foundation Graduate Research Fellowship, and United States Air Force Contract F30602-00-2-0501. A preliminary

## 1 Introduction

The network resources in the Internet are dynamically shared among a large number of users, posing a significant challenge in the guaranteed provisioning of quality-of-service (QoS) to individual users. During the last several years, QoS issues in the Internet have attracted significant research interest as well as commercial investments. One of the ways to achieve QoS guarantees on a per-flow basis is to make *a priori* reservations of buffer and bandwidth resources in the network. This approach is used in the Integrated Services (IntServ) architecture [4], which relies upon a reservation setup protocol such as RSVP [26]. The per-flow management required at the routers, however, calls into question the scalability of this approach. The Differentiated Services architecture (DiffServ) [3] is an alternate method that achieves improved scalability by aggregating data packets into a small number of service classes and defining router behaviors expected by packets belonging to each of these classes.

DiffServ allows up to 64 different service classes that serve only to define the treatment a packet will receive in relation to other packets, but do not provide absolute guarantees on performance. In the absence of guarantees, as in IntServ, the role of capacity planning for traffic from various classes of service becomes critical to achieving satisfactory service. These bandwidth contracts, referred to as service-level agreements (SLAs), can provide reasonable guarantees only when established over long time scales [18]. The user demands for various levels of service can change rapidly due to a variety of reasons; participation in SLAs between providers, therefore, is not likely to lead to an efficient use of network resources. Mechanisms for capacity planning and congestion control through dynamic pricing, however, can be significantly more efficient and also more responsive to changes in, and the demand for, the network resources. This paper explores a practical, flexible and computationally simple user-centric pricing strategy that can achieve QoS provisioning in DiffServ networks with multiple priority classes operating in an efficient economic market, while also maintaining stable transmission rates from end-users.

### 1.1 Related Work and Motivation

Over the last several years, a number of research proposals have advocated the creation of an economic market for congestion control and differentiated services. One of the simplest means of achieving QoS differentiation is through Paris Metro Pricing [20], which assigns fixed prices for traffic from different service classes and logically partitions the network into separate channels, each

---

version of this paper appeared in *Proceedings of the IEEE International Conference on Communications*, April-May 2002, New York, NY, pp. 986–990.

with its own allocated bandwidth. All packets traveling through the same channel receive the same level of best-effort service; however, channels with higher prices provide better service because they attract less traffic. Such a scheme leads to an inefficient utilization of the network resources since bandwidth allocated to the higher-price channels may go unused.

An alternate method, which does not pre-allocate bandwidth to different service classes, is based on dividing traffic into multiple priority classes and using a different, but fixed, price associated with each service class [6, 17]. Packets belonging to higher-priority classes are given greater forwarding priority at the intermediate routers, thus leading to QoS differentiation. Such a scheme is rendered more efficient by the addition of a congestion-dependent component to the price, allowing for a more competitive price to be offered. One such scheme is studied in [1], where the pricing changes at a time-scale suitable for typical human response times. In this system, a technical (non-economic) method of congestion control is required at shorter time-scales. The absence of an economic congestion control mechanism at short time-scales, however, may expose the system to abuse by software application vendors or sophisticated users.

Another scheme utilizing congestion-dependent pricing is studied in [22], where the charges are determined on a per-call basis and are assessed at the time the call is admitted. However, rapid variations in the demand within the duration of one call, typical of Internet traffic today, render such an approach less

the price for each service class depends on the average demand for that service class and is negotiated for short intervals of time. This requires a resource negotiation protocol so that the network can commit resources for these short durations. However, a user may incorrectly anticipate his/her requirements and request a resource commitment but not actually use it. Therefore, such reservation-based approaches, even for short durations, can lead to higher charges for users and inefficient use of the network. In a somewhat related pricing strategy [7], resources may be provisioned per service class over the long term and then priced based on user demand over the short term. This can lead to improved utilization of the network resources over the short term as users dynamically change transmission rates and/or switch service classes. Such provisioning may be engineered to maximize profits as shown in [7]; however, over the long term, static allocation of bandwidth among the service classes will lead to a less efficient market with sub-optimal long-term pricing and resource utilization.

An altogether different principle is used in another set of approaches based on the “smart-market” proposal by MacKie-Mason and Varian [15], where packets are each marked with a bid price that reflects the need of the sending user for the packet to be transmitted. The router admits packets whose bid

price is greater than a certain cutoff amount, which is in turn a function of the congestion state of the router. The originator of the packet is charged the lowest bid of all the packets admitted to the router during the time period. Gibbens and Kelly [9] also describe a price-based feedback system for congestion control where packets utilizing a congested router's services are marked as such and end-users are assessed a charge based upon the number of packets thus marked. This principle is extended in [2] and other related works, where a price variable maintained at each router represents the measure of congestion and is used to determine the probability with which a packet is marked at the router. The marking probability at each router is exponential in the current price, ensuring that the end-to-end probability that a packet is marked before it reaches its destination accurately captures the congestion along the path of the packet. As in [9], users may be charged based on the number of packets that are marked. A further extension of such a pricing scheme, which uses more than one bit in the packet for the price feedback, is discussed in [8]. In this pricing strategy, a value representing a price is assigned to each packet as it enters the router. This price is an expression of the social cost incurred by other users due to congestion. Users adapt their transmission rates based on the recent history of prices marked on the packets.

All of the above mentioned pricing schemes provide for economic regulation of individual bandwidth consumption but do not facilitate QoS provisioning for multiple levels of priority service. They all rely upon an FCFS discipline for packet scheduling at the routers and therefore, do not provide a means for one customer to obtain priority service over another. For example, a customer requiring very low delays but not much bandwidth cannot obtain the desired service when pricing schemes are based on the bandwidth consumed but not on the scheduling services rendered.

A pricing scheme that seeks to provide multiple levels of service, with price computations based on both the flow rates and the waiting times, is discussed in [11]. The computation of prices in this strategy is done periodically using an iterative approach in which the history of flow rates and waiting times determines current price estimates. In this scheme, a customer evaluates the current advertised price information, and then chooses how to transmit its packets (such as priority level and flow rate). Thus, the prices charged to a customer may not reflect the true cost of the service provided since the prices are based on the predicted level of service for each priority class as opposed to the actual service rendered. For example, the charge for a high-priority class may be high even if its packets do not actually receive priority service (such as when there is no lower priority traffic). As a related consequence, this scheme does not provide a framework to define the relationships between the different prices for the priority levels based on current usage patterns within each priority level. Thus, the prices set for the different priority levels and the user actions based on these prices will not represent an efficient market

and therefore, may not lead to efficient management of the available capacity in the network. Additionally, the scheme specifies that the nodes which price the packet also perform a running average of the state of the network usage. Besides imposing complexity on the network core instead of the network edges, this limits the amount of information available to the user regarding the burstiness of the current network utilization. We feel that such averaging should be performed at the end-systems to permit the user to determine its own method of adapting to the fluctuations in the prices.

## 1.2 Contributions

In this paper, we propose a practical, flexible and computationally simple pricing strategy that can achieve QoS provisioning in Differentiated Services networks with multiple priority classes while maintaining stable transmission rates from end-users. Our goal in this paper is to improve efficiency of network resource usage and reduce router processing overheads by avoiding resource reservations and static allocations in the network for QoS provisioning. Most importantly, in contrast to previous work in pricing the Internet, our goal is to achieve efficiency in market economics by considering the *actual* demand for each resource in the determination of prices charged for the use of the resource. For example, a high-priority packet that arrives at a router should not be charged a higher price if there are no lower-priority packets in the router over which it receives preferential service.

This paper borrows the framework described in [8], but uses an altered pricing strategy to achieve differentiated services through dynamic pricing based on the principles stated above. With simple queuing and scheduling policies in the routers, our pricing strategy is able to provide for varying levels of scheduling priority, and is ideally suitable for DiffServ networks. The strategy, which allows for a flexible and appropriate gradation in the pricing for different priority classes, involves charging packets based on,

- (1) The bandwidth consumed and the instantaneous demand for the bandwidth.
- (2) The preferential service *actually* received by the packet in the allocation of the bandwidth as well as the buffer resources.

The pricing strategy does not assume a prior contract between the network and the customer for a certain QoS/price combination, but instead relies on user adaptation of the transmission rate and/or the service class based on the price feedback it receives for each packet. In this paper, when we refer to the “user”, we do not necessarily mean a human end-user. The user may be a human end-user, an end-user application software, a service provider or a

border router in a DiffServ domain.

A number of other approaches in the research literature propose pricing strategies that either maximize revenue or maximize social welfare. For reasons described in [23], our approach does not follow this optimality paradigm and seeks to address the more practical and architectural issues, while creating a pricing strategy based on an efficient market. In addition, one may argue that our strategy uses a stronger user-centric approach since the price charged for each transmitted packet is based on the preferential service *actually* rendered to the packet and not based on the service expected by the user prior to the transmission of the packet. Such a scheme also leads to a more efficient utilization of the network since it avoids static allocation of network resources either on a per-flow basis or on a per-class basis.

Our strategy is computationally simple and practical for easy implementation in switches and routers, and is scalable with an  $O(1)$  complexity with respect to the number of flows. The queue architecture within routers assumed by our pricing and scheduling strategy is similar to those in DiffServ-capable routers. This paper does not address the various logistical and philosophical objections that may exist to usage-sensitive pricing of network services. A treatment of these topics may be found in [14].

### 1.3 Organization

The rest of this paper is organized as follows: Section 2 describes the router and the system model used in our analysis. This section also describes the assumptions, requirements and goals of this work. Section 3 describes our pricing strategy and the rationale behind it. This section also presents a pseudo-code implementation of the pricing strategy. Section 4 presents a queuing-theoretic analysis of the stability of the pricing function through obtaining the buffer distribution that results from use of such a pricing function. Section 5 presents simulation-based results which further demonstrate that, with user adaptation in combination with our pricing strategy, differentiated services and stable transmission rates can be achieved. Section 6 concludes the paper with a discussion of various operational issues associated with actual deployment of the pricing strategy.

## 2 System Model: Assumptions and Goals

Our system model consists of a network of packet-switched routers, connecting together a *finite* number of end-systems, or hosts. Each host is a source and/or

a sink of data packets. Every data packet carries a field in its control header which represents the price that will be charged for its transmission. Each router in the path of a packet adds its charge to this field in the packet. When the packet reaches its destination host, this price field is copied into its acknowledgment and returned back to the sending host. The price field in the packets may be best accommodated either as a separate IPv4 option or as a new sub-header to be encapsulated in an IPv4 packet. The user may employ the recent history of prices charged to dynamically adapt and adjust its sending rate or select a different class of service. Our goals in developing a pricing strategy are as follows.

- *Pricing based on cost and demand:* The architecture should ensure that the price of any given network resource (bandwidth, buffer or preferential service) depends only on the current demand for the resource and the cost to the provider. For example, in the absence of packets from lower priority classes, the demand for preferential service does not exist, and the price charged to a high-priority packet should be no more than that charged to a lower priority packet that encounters a similar router state. Pricing based on cost and demand leads to a more efficient market with more competitive prices.
- *Scalability and Computational Simplicity:* The pricing strategy should be simple and easily implementable in switches and routers. In the spirit of the original motivation behind DiffServ, there should be no per-flow management at the routers. An  $O(1)$  computational complexity, with respect to both the number of flows and the number of packets buffered in the router, is desirable of all pricing and scheduling algorithms implemented at each router.
- *Dynamic in-session user adaptation:* Internet traffic characteristics display significant variations over short and long durations of time. Therefore, the demand for a resource is likely to change during the lifetime of an active flow. Pricing should reflect the current demand, and the user should be able to adapt its transmission rate at any time during the lifetime of the session.
- *Stability:* While any real control system with delayed feedback will generate some oscillatory behavior, the pricing framework should not coerce users into an unstable or chaotic traffic pattern. It should be possible for a user to transmit at a rate that is stable while maximizing its consumer surplus.

The framework described in [8] along with our pricing strategy meets the goals stated above, as will be shown in subsequent sections.

Let  $Q$  be the total number of classes of service, labeled  $q \in \{1, \dots, Q\}$ , in order of increasing priority. Packets that arrive at a router and are headed to a certain output link are all queued separately based on the class of service to which the packet belongs. We define a *queue* as a logical entity containing a sequence of packets from the same service class that have to be served in a



FIFO order. The queue architecture assumed is a simple one similar to that in DiffServ-capable routers. In fact, such a queue architecture with the capability to classify packets based on their service classes and treat them accordingly have also been implemented in simple hardware switches. Note that, depending on the buffering architecture of the switch or the router, a queue may not be the same thing as a buffer since a single buffer can implement multiple logical queues [24]. We label the queues by the service class of packets in them, i.e., packets of service class  $q$  await transmission in queue  $q$ .

In our system model, the router uses a simple scheduling algorithm that gives non-preemptive priority to a higher service class in scheduling packets for transmission through the output link. Also, when the buffer is full and a packet arrives from a service class  $q$ , the necessary number of packets from the service classes below  $q$  are dropped to make room for the new packet. Packets from the lowest class are dropped first. If the buffer is full and there are no packets in it belonging to a lower service class, the new packet is dropped.

### 3 Pricing Strategy

Denote by  $N_q(t)$  the total number of bytes of data at time instant  $t$  in queue  $q$  belonging to packets that have fully arrived into the queue, but have not yet begun transmission through the output link. Consider a packet of length  $l$  and service class  $S$ . Let  $t_A$  denote the time at which this packet completely *arrives* at one of the router queues. Let  $t_B$  denote the time instant the packet *begins* transmission through the output link.

We now describe the three components of the price charged to each packet in our scheme. The sum of these price components is the total charge billed corresponding to the packet.

*Price due to bandwidth consumed:* This component of the price is a function of the length of the packet and the current demand for bandwidth on the link. At the instant that a packet begins transmission, the set of packets that have completely arrived at the router and are awaiting transmission are interpreted as the total demand for bandwidth at that instant. In most Internet router architectures, the price charged to the packet can be stamped on it only before it begins transmission. Therefore, we ignore the demand due to packets that become available for transmission after time  $t_B$ . We expect this approximation to have negligible effect on the dynamics of this scheme.

Let  $f_{bw}(x)$  be the price-demand function expressing the price per unit of bandwidth consumed when the demand for the bandwidth is  $x$ . Using the above mapping between the number of bytes awaiting transmission and the demand

for bandwidth, one can now express this component of the price assigned to the packet as,

$$P_{bw} = lf_{bw} \left( \sum_{q=1}^S N_q(t_B) \right) \quad (1)$$

The assigned price is equal to the best-effort price (or the access charge) when the demand is zero. Note that, for ease of implementation as well as to ensure that the feedback to the user is based on the most current state of the router, we use the size of the queue at time  $t_B$  instead of time  $t_A$  to compute this component of the price.

*Price due to preferential service rendered:* This component of the price is based on the preferential service received by the packet, defined in terms of the number of other packets over which it receives priority. Let  $f_{ps}$  be the price-demand function for the preferential service received by a packet when the demand for this preferential service is  $x$ . A packet in service class  $S$  gets ahead of all the packets in queues of classes below its own, since the router serves packets from the highest class first. The total size of data over which a packet receives priority is the number of bytes of data in the queues corresponding to service classes below it. In our pricing strategy, this quantity is interpreted as the demand for the preferential service enjoyed by the packet. Therefore, the price due to preferential service charged to a packet that arrives fully at time  $t_A$  is given by,

$$P_{ps} = lf_{ps} \left( \sum_{q=1}^{S-1} N_q(t_A) \right) \quad (2)$$

While the above two components of the price appear similar, they are both necessary to ensure a pricing strategy that clearly recognizes the distinction between the demand for the bandwidth resource and the demand for preferential service by the router. For example, consider a router state with a large number of packets in its buffers, all of the same service class  $q$ . If we did not use a separate price component for preferential service, and if there were no packets of higher class in the queues, a new packet in a higher class could be served ahead of the large number of packets in the queue  $q$  and still be charged only approximately the same amount as the packet at the head of the queue  $q$ . A more efficient market would ensure a fairer price where gaining priority over a large number of other packets costs more than gaining priority over a smaller number.

In addition, it is also possible to argue that the pricing component due to preferential service helps reduce the length of traffic bursts on the network. A simple example would be two traffic streams of separate priority classes,  $S_1$

and  $S_2$ , in a non-preemptive queue. Without a price component for preferential service, a traffic burst utilizing the higher-priority service class,  $S_2$ , could block traffic of the lower-priority class,  $S_1$ , for a long duration of time. This would force either a long burst of class  $S_1$  after the completion of the class  $S_2$ 's burst, or worse, a buffer overflow in the  $S_1$  class. With pricing based on preferential service rendered, however, departing traffic of class  $S_2$  will carry pricing information indicating the depth of  $S_1$ 's queue and will lead to a reduction in the sending rate of traffic class  $S_2$ . In effect, this shortens the size of the burst from  $S_2$ , reduces the amount of time  $S_1$  is starved of service and in addition, shortens the burst of traffic class  $S_1$  as the queue clears.

This separate pricing component for the preferential service delivered also helps capture other parameters, such as delay, not captured by the congestion state and bandwidth consumed. The above two components of the price assigned to a packet differ in the following two ways:

- The price assigned for bandwidth consumed depends on the state of the router at the instant that the packet begins transmission, while that for preferential service rendered depends on the router state at the instant the packet fully arrives at the router.
- The price assigned for bandwidth consumed depends on the number of bytes of data in the queue of the packet's own service class, while the price assigned for preferential service does not.

*Pricing due to buffer resources occupied:* This third component of our pricing strategy is intended to reflect the cost of buffer resources occupied by a packet and the packet losses incurred by other flows due to it. The price assigned to a packet due to the buffer resources occupied by it depends only on the number and sizes of the packets that were denied this buffer space due to its occupation of the space. This is the sum of the sizes of the packets that are dropped during the time interval between the instant that a packet has arrived at the router and the instant it begins transmission. Let  $D_q(t_1, t_2)$  be the number of packets of service class  $q$  that are discarded during the time interval  $(t_1, t_2)$ . The price assigned to a packet of length  $l$  and service class  $S$  due to the buffer resources occupied by it is, therefore, given by,

$$P_d = lf_d \left( \sum_{q=1}^S D_q(t_A, t_B) \right) \quad (3)$$

where  $f_d(x)$  is the corresponding price function with regard to buffer occupancy.

Fig. 1 shows a pseudo-code implementation of the above pricing strategy. In the pseudo-code,  $\bar{N}(q)$  expresses the sum of the sizes of all the packets that have fully arrived in queues below and including  $q$ .  $\bar{D}(q)$  expresses the number of

*Initialize:* (Invoked when the router is initialized)

```

begin
  for ( $i = 1; i \leq Q; i = i + 1$ )
     $\bar{D}(i) = 0;$ 
     $\bar{N}(i) = 0;$ 
  end for;
end;

```

*Enqueue:* (Invoked upon the arrival of a packet)

```

begin
  if PacketIsDropped
    for ( $i = Pkt.ServiceClass + 1; i \leq Q; i = i + 1$ )
       $\bar{D}(i) = \bar{D}(i) + Pkt.Size;$ 
    end for;
  else
    for ( $i = Pkt.ServiceClass; i \leq Q; i = i + 1$ )
       $\bar{N}(i) = \bar{N}(i) + Pkt.Size;$ 
    end for;
     $Pkt.Price = Pkt.Price$ 
       $+ Pkt.Size \times f_{ps}(\bar{N}(Pkt.ServiceClass - 1));$ 
     $Pkt.IncomingDropCount = \bar{D}(Pkt.ServiceClass);$ 
  end if;
end;

```

*Dequeue:* (Invoked upon scheduling a packet for transmission)

```

begin
   $DropCount = \bar{D}(Pkt.ServiceClass)$ 
     $- Pkt.IncomingDropCount;$ 
   $Pkt.Price = Pkt.Price$ 
     $+ Pkt.Size \times f_{bw}(\bar{N}(Pkt.ServiceClass));$ 
   $Pkt.Price = Pkt.Price + Pkt.Size \times f_d(DropCount);$ 
  for ( $i = Pkt.ServiceClass; i \leq Q; i = i + 1$ )
     $\bar{N}(i) = \bar{N}(i) - Pkt.Size;$ 
  end for;
end;

```

Fig. 1. Pseudo-Code Implementation of the Pricing Strategy

packets that have been dropped from classes below  $q$ . For purposes of brevity, the pseudo-code ignores the issue of the number  $\bar{D}(q)$  wrapping around in a variable with a finite number of bits.

We assume that the three price functions involved,  $f_{bw}()$ ,  $f_{ps}()$  and  $f_d()$ , are either simple exponential functions or can be readily determined by the router for any given value through a table look-up. This allows a flexible pricing strategy through the ability to tune these functions as needed. To reduce memory requirements, one may implement these as step functions at the cost of a small amount of loss in pricing accuracy.

## 4 Pricing Function and Stability

We have chosen to examine the stability of our pricing strategy from a queuing theoretic standpoint. Modeling our scheme in a discrete mathematical framework provides several advantages over a rate-based model often used in a game theoretic framework, such as those recently explored by Cao *et al.* in [5].

- Requirements for stability in Markovian systems have been rigorously defined in the literature [13] and accepted by the networking community.
- It is possible to derive closed form expressions which relate a pricing scheme to real world parameters, such as the probability of packet loss due to router buffer overflow.
- Since routers perform scheduling and forwarding tasks on a per-packet basis, as also briefly argued in [1], real router behavior in packet-switched networks is best modeled using a queuing theoretic framework. Flow-based frameworks model the system in terms of continuous values and rates, which tend to approximate rather than accurately capture the true state of the router.

Our definition of stability for the proposed pricing structure is an extension of Kleinrock's results in [13] to encompass our economic model. We state that a pricing function is both *stable* and *feasible* if and only if the capacity of the network is sufficient to satisfy the traffic demand required by users who are each attempting to maximize their individual consumer surpluses. Recall that the consumer surplus represents the difference between the utility achieved by the consumer (as measured by the maximum price the consumer is willing to pay) and the actual price the consumer pays. As discussed in [13], a finite value of the expected queue length implies stable operation.

We choose to examine the polynomial class of pricing functions utilized in [8] and [21], which take the form shown in Equation (4) below.

$$P(n(t)) = \left( \frac{an(t)}{C} + b \right)^k \quad (4)$$

The above function computes a price  $P(n(t))$  based upon the current buffer occupancy  $n(t)$  as a function of the total buffer capacity  $C$  and constants  $a$ ,  $b$ , and  $k$ . As shown in subsequent analysis, careful selection of these constants leads to stable network behavior. There are several characteristics inherent to the polynomial pricing function which make it preferable to other schemes explored in the literature, such as the barrier pricing function examined in [19] and provided below:

$$P(n(t)) = a \left( \frac{1}{C - n(t)} - \frac{1}{C} \right)^{\frac{1}{b}} \quad (5)$$

These advantages of a polynomial pricing function include:

- The price stamped by a barrier function asymptotically approaches infinity as the buffer approaches its full capacity. Since it is possible for a burst of traffic to arrive from several independent sources, each of which has no prior knowledge of the buffer state, it is also possible for a user to be unfairly penalized for sending only one packet.
- Polynomial pricing functions allow for the network engineer to define a maximum price that can be charged to a packet requesting service. Allowing customers to know that such a maximum price exists for each service class will ease market acceptance.

The barrier pricing function, with prices approaching infinity as the use of network resources approaches full capacity, can be readily seen to be stable. In the following, we show that the polynomial pricing function can also provide stable transmission rates for the end-users.

#### 4.1 *Queuing Analysis*

We model the system as an  $M/M/1$  queue where the total amount of traffic handled by the network is the sum of the traffic submitted by multiple users operating independently of one another. Each user seeks to maximize its own consumer surplus, balancing its utility gained from the network services against the price it has to pay for the service.

As in [8] and [21], we assume that each user  $i$  has a predefined utility function of the form,

$$u_i(\lambda_i(t)) = \begin{cases} w_i \frac{\lambda_i(t)^\beta - 1}{\beta} : \beta < 1 \\ w_i \log(\lambda_i(t)) : \beta = 0 \end{cases} \quad (6)$$

The variable  $w_i$  may be interpreted as the user's need for bandwidth, and consequently, the willingness to pay for said traffic. In the above equations,  $\beta$  represents the steepness of the demand curve as a function of bandwidth  $\lambda_i(t)$ . In the following, for brevity and clarity of presentation, we consider only the case where  $\beta = 0$ .

A price, calculated by the polynomial function discussed earlier, is affixed to each packet. If we assume that feedback is instantaneous, it then becomes the goal of each user to maximize its individual consumer surplus,  $s_i$ :

$$s_i(\lambda_i(t)) = u_i(\lambda_i(t)) - \lambda_i(t)P(n(t)) \quad (7)$$

$$\lambda_i(t) = \min \left\{ \frac{w_i}{P(n(t))}, \lambda_i^{max} \right\} \quad (8)$$

where  $\lambda_i^{max}$  is the maximum possible transmission rate of node  $i$ . If we establish a desired rate for each user, and the total desired rate of each user is within the bandwidth capacity of the router, the unconditional stability requirement is met. In this case, users can ignore short term price fluctuations in much the same way that commuters ignore small jumps in the price of gasoline.

Assuming that each user is continually attempting to maximize its own consumer surplus, it can be shown that, through combining Equations (4) and (8), the traffic submitted by each user for transmission is given by:

$$\lambda_i(t) = \min \left\{ \frac{w_i}{\left( \frac{a(n(t)+1)}{C} \right)^k}, \lambda_i^{max} \right\} \quad (9)$$

The queue state probabilities and the expected queue size of a system operating under the pricing function presented in (4) are shown below:

$$p_n = \begin{cases} \frac{(\mathcal{F}(a,k))^n \frac{1}{n!k}}{1 + \mathcal{F}(a,k)_1 F_k \left[ \frac{1}{2_1, \dots, 2_k}; \mathcal{F}(a,k) \right]} : k > 1 \\ e^{-\mathcal{F}(a,k)} (\mathcal{F}(a,k))^n \frac{1}{n!} : k = 1 \end{cases} \quad (10)$$

$$E[n(t)] = \begin{cases} \frac{\mathcal{F}(a,k)_0 F_{k-1} \left[ \frac{1}{2_1, \dots, 2_{k-1}}; \mathcal{F}(a,k) \right]}{1 + \mathcal{F}(a,k)_1 F_k \left[ \frac{1}{2_1, \dots, 2_k}; \mathcal{F}(a,k) \right]} : k > 1 \\ \mathcal{F}(a,k) : k = 1 \end{cases} \quad (11)$$

where

$$\mathcal{F}(a,k) = \left( \frac{C}{a} \right)^k \frac{\sum_i w_i}{\mu} \quad (12)$$

and  ${}_nF_m[\dots; x]$  is a simplified notational form of the generalized hyper-geometric function [12]:

$${}_nF_m \left[ \begin{matrix} a_1, a_2, \dots, a_n \\ b_1, b_2, \dots, b_m \end{matrix}; x \right] = \sum_{k=0}^{\infty} \frac{(a_1)_k (a_2)_k \dots (a_n)_k}{(b_1)_k (b_2)_k \dots (b_m)_k} \frac{x^k}{k!}$$

$$(a)_k = \frac{\Gamma(a+k)}{\Gamma(a)}$$

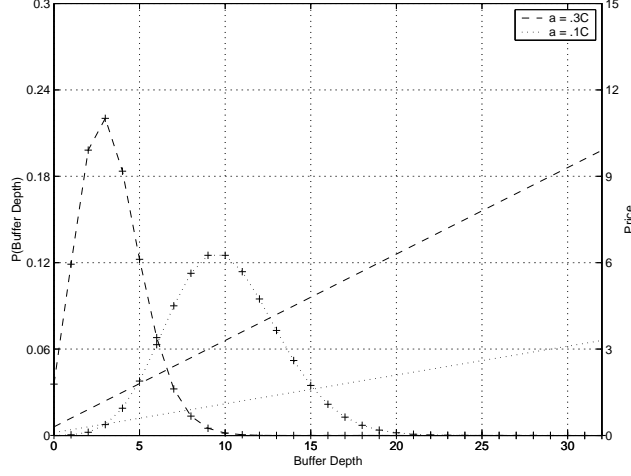


Fig. 2. Buffer Distribution under Polynomial Pricing Function

The reader is referred to the Appendix for a full derivation of the above result. Note that, for  $k = 1$ , the expected queue length resolves to the mean of the Poisson process.

#### 4.2 Parameter Determination

As stated earlier, a pricing function is stable and feasible when the optimal traffic load generated is within the capacity of the network. The operational capacity of the network, however, is a function of the applications utilizing the bandwidth and may depend on such factors as mean buffer occupancy and packet loss rate. The multiple parameters available in the pricing functions discussed above provide the network engineer with the tools needed to perform such traffic shaping.

In general, the service provider chooses a targeted packet drop probability,  $p_d$ , and an expected buffer occupancy,  $E[n(t)]$ , as shown below:

$$p_d = \sum_{n=C+1}^{\infty} p_n \quad (13)$$

$$E[n(t)] = \sum_{n=0}^{\infty} np_n \quad (14)$$

Fig. 2 shows the effects of manipulating a single pricing function on the buffer distribution. The expected buffer occupancy is displayed as +-marked curves, while the straight lines represent the polynomial pricing functions. The buffer occupancy shifts from 10 packets to 3.3 packets as the parameter  $a$  changes from  $0.1C$  to  $0.3C$ . It is possible to further limit the buffer overrun probability



by increasing  $k$ . This, in turn, causes the price affixed to the packet to rise at a greater rate with respect to the buffer occupancy.

## 5 Simulation

### 5.1 Price-Demand and Utility Functions

For the purpose of simulation, we model a collection of individual users who continually optimize their individual bandwidth allocation according to the utility functions defined in Section 4.1.

In a real world situation, the assumption that a user will have instantaneous knowledge of the price state of the router is not applicable. Therefore, the user needs to compute an estimated price based solely upon the history of prices stamped on its packets. Let  $\bar{p}(t - \tau, t)$  denote the average value of the price stamps on packets received within the time interval  $(t - \tau, t)$ . The estimated price at any given instant  $t$  is the weighted average of the previous price estimate and  $\bar{p}(t - \tau, t)$ . In our simulations,  $\tau$  is set to a value of approximately two round-trip times.

$$\hat{p}(t) = (1 - \alpha) \hat{p} + \alpha \bar{p}(t - \tau, t) \quad (15)$$

Unlike in [8], a constant feedback rate is not assumed or needed for each user to determine their needed bandwidth. The target bandwidth shown in Equation (7) is used to determine the probability that a user will transmit a packet during any given cycle.

As in other related works, we use an iso-elastic price curve with respect to the demand, as follows:

$$P_{bw} = \left( \frac{a_{bw}}{C} \sum_{q=1}^S N_q(t_B) + b_{bw} \right)^{k_{bw}} \quad (16)$$

where  $a_{bw}$  is a function of the total buffer size,  $b_{bw}$  is a function of the desired maximum buffer occupancy, and  $k_{bw}$  defines the steepness of the price-demand curve as the demand increases.

As in the case of bandwidth consumed, we assume an iso-elastic price curve

defining the preferential service component of price as well, given by,

$$P_{ps} = \left( \frac{a_{ps}}{C} \sum_{q=1}^{S-1} N_q(t_A) + b_{ps} \right)^{k_{ps}} \quad (17)$$

where  $a_{ps}$ ,  $b_{ps}$ , and  $k_{ps}$  serve similar curve-shaping functions as in  $P_{bw}$ . The price curve for buffer resources is similarly given by,

$$P_d = \left( \frac{a_d}{C} \sum_{q=1}^S D_q(t_A, t_B) + b_d \right)^{k_d} \quad (18)$$

## 5.2 Simulation Environment and Results

We simulate ten users with different utility-demand curves competing for service at a single router. The router can transmit only one packet per cycle, while a total of eight packets can be submitted to the router per cycle. The size of the shared buffer,  $C$ , is set to carry a maximum of 50 packets, with each packet being 576 bytes in size. Each of our simulation experiments runs for 30,000 cycles. The simulated pricing functions are chosen as follows.

$$P_{bw} = \left( C^{-1} \sum_{q=1}^S N_q(t_B) + 1 \right)^2 \quad (19)$$

$$P_{ps} = \left( 8C^{-1} \sum_{q=1}^{S-1} N_q(t_A) \right)^4 \quad (20)$$

$$P_d = \left( 16C^{-1} \sum_{q=1}^S D_q(t_A, t_B) \right)^4 \quad (21)$$

In the first of our simulation experiments, eight users send traffic in the lowest available service class. Each individual user, however, has a slightly different utility function. As can be seen from Fig. 3, all eight users realize a steady-state convergence to an optimum bandwidth over the long term. While the figure appears to show an increased jitter in the transmission rates for high-bandwidth users, these variations as a fraction of the average transmission rate are approximately the same as in the low-bandwidth users. Users who demand the most bandwidth, and therefore send more packets, receive price feedback more often than the low-bandwidth users. The user adaptation function of a high-bandwidth user, therefore, has more reliable information about the recent history of router state and adapts more optimally. This effect, however,

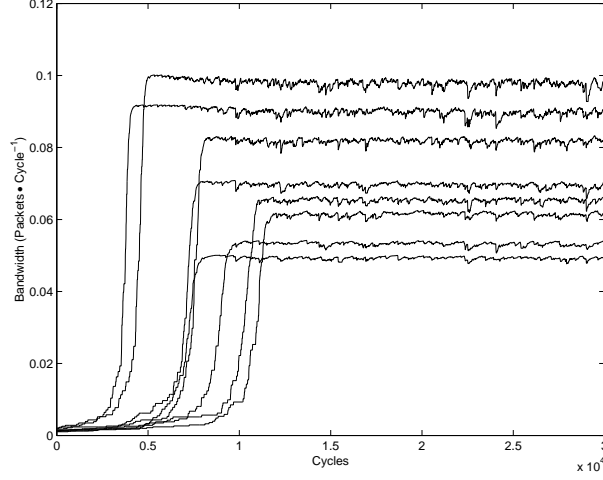


Fig. 3. Bandwidths achieved with all users in the same service class

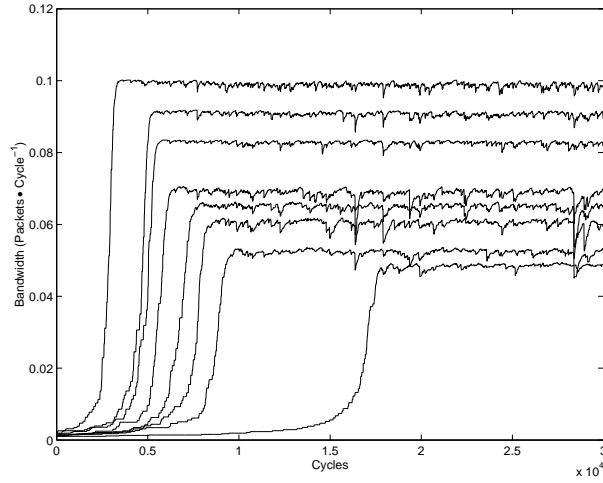


Fig. 4. Bandwidths achieved with each user in a different service class

is countered by the fact that high-bandwidth users suffer from an increased average delay since they are more likely to continue sending packets when the buffer occupancy is high. This introduces an undesired delay in the control loop for high-bandwidth users, and allows larger oscillations in the transmission rates.

In the second of our experiments, we re-examine the situation with each user sending traffic in a different service class. As shown in Fig. 4, with each of the users having an infinite number of packets available for transmission, the higher-priority users demand successively lower amounts of traffic. This is a commonly cited example of differentiated service network utilization, where large scale bulk traffic use, such as e-mail, is relegated to a lower priority queue, and high priority traffic, such as key frames in an MPEG, is placed in a higher scheduling priority. Also, as can be seen in Fig. 5, the higher the user's desired service class, the higher the price and the lower the average

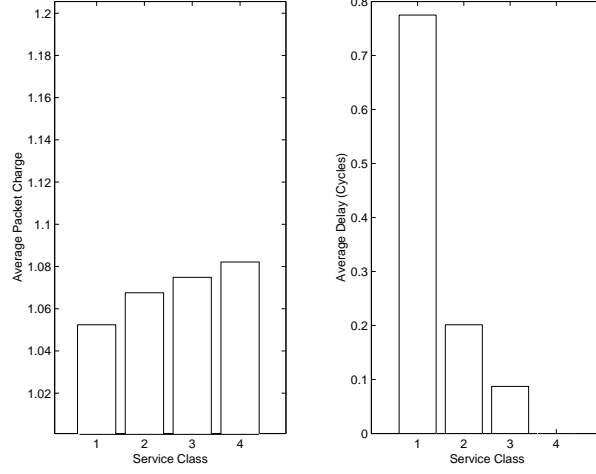


Fig. 5. Average price and delay for each class

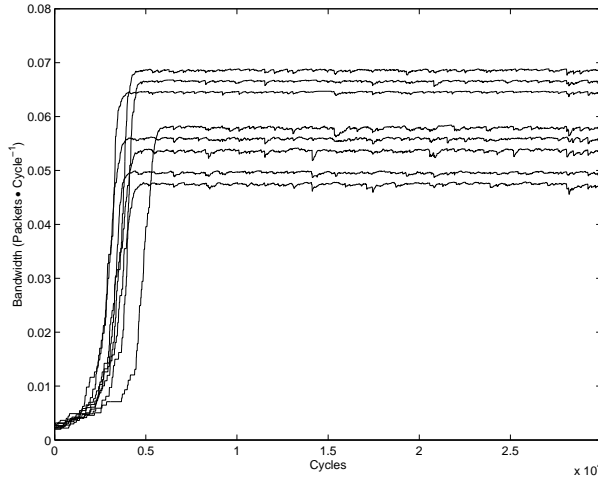


Fig. 6. Bandwidths achieved with class differentiation across a multi-hop topology

delay experienced.

The third experiment we conducted dealt with performance across multi-hop topologies. Rather than passing through a single bottleneck device, the traffic generated passes through a pair of routers, each adding its own charge for preferential service and buffer occupancy. As can be seen in Fig. 6, even though the users are operating in multiple priority classes, the overall stability of the user's bandwidth demands has improved. The price stamped on a packet that traverses a mutlihop topology is smoothed by the stochastic nature of the traffic that is not part of, but intersects with, the flow as it traverses the multiple routers.

## 6 Operational Issues and Concluding Remarks

There are several different ways to implement the pricing strategy discussed in this paper in a large-scale network such as the Internet. In the following, we briefly discuss some operational issues along with a feasible multi-phase approach to the actual deployment of the pricing strategy.

### 6.1 *Availability of best-effort service*

Critics of economic approaches to congestion control and preferential service argue that the Internet is free and should always remain so. It is easily argued that the Internet is not actually free [16], but the authors do agree with the sentiment behind the critics' argument: best-effort traffic should not be charged beyond the charge for network access. In a preferential service scheme regulated through pricing, best-effort traffic is essentially a background event that would have little bearing on the performance of paying customers. The pricing functions presented should be configured so that the only packets accounted for are those requesting some level of service over that of best effort. It is in this fashion that dynamic pricing such as that presented in this paper, along with the automated provisioning behavior associated with it, will allow for best-effort traffic to peacefully co-exist alongside preferential-service traffic.

### 6.2 *Initial deployment*

An initial roll out of a pricing strategy can occur between ISP peers who are observing a mutual SLA for the purpose of QoS provisioning through DiffServ. In this situation, both ISPs are already performing traffic metering at the endpoints of the network and have entered into a business agreement that places a monetary value upon real network traffic parameters, such as the bandwidth utilized and the percentage of packets dropped. Rather than place prices on the bulk volume of traffic exchanged in each service class, the two network peers can agree on a dynamic pricing strategy which would serve to limit the number of packets dropped, perform traffic shaping at each endpoint and serve to continually account for the amount of traffic passed by the peers across each other's borders.

A multi-homed ISP has even more options for developing and manipulating pricing contracts. Rather than paying a conventional flat rate or a per-bit price for access from its upstream providers, such an ISP can negotiate for a dynamic, congestion-dependent pricing strategy with both upstream ISPs. The

downstream ISP can then utilize a constraint-based routing algorithm locally to minimize cost for its own outbound traffic, and then pass the savings on to its customers, thus leading to a fair and almost perfect market for bandwidth.

Since each router in the path of a packet adds its charges to the price field in the packet, it may be impossible to identify how much of the total charge assessed for each packet originates from which ISP in the traversed path. Therefore, the billing for the total charge can only come from one source which will most likely be the user's ISP. However, each ISP would have to accumulate all its charges for all packets from each of its upstream ISPs and bill them accordingly. Thus, while the user pays its own ISP, each of the downstream ISPs, through bilateral arrangements, will need to recover their costs from their respective upstream ISPs. The billing chain between the ISPs will ensure that each ISP is paid for exactly the amount assessed to each packet it forwards, even though the packet itself does not carry a breakdown of the price information among the different ISPs.

### *6.3 End-user integration*

From the end-user's point of view, when the adaptation in our pricing strategy is accomplished at an intermediate point such as the ISP, using a congestion dependent pricing network will be no different from using a conventional network. For example, the local ISP can perform dynamic traffic shaping and manage the continual flow of pricing information without informing the end-user of the congestion states of the network. Rather than relying on the end-user to continually update his or her transmission rate based upon the latest congestion information, the local ISP can establish congestion-dependent price points for different levels of service. These price points can be updated hourly and passed on to the user via a local status web page, or can be updated monthly, so as to allow for the user to sign a flat-rate contract that encompasses a variety of network services.

However, there are certain advantages to pushing the pricing information to the end-user; most importantly, it allows the individual end-user or the end-user application greater freedom in the control of its own network connection and the associated traffic it generates. The incentive to regulate the traffic will derive from the dynamic pricing information conveyed to it. While it can be argued that the volume of pricing information presented to the end-user is overwhelming and unusable on the time scales corresponding to human response times, the end-user software may facilitate human decision-making by extracting trends in the pricing information and presenting them to the human user in a convenient format at appropriate intervals of time. Alternately, a more effective technique would be for the end-user application software to

adapt based on *a priori* instructions by the human end-user. There is a long precedence for such actions taken by end-user software. Conventional congestion control schemes, such as those employed by TCP, already take place without the user's knowledge or direct intervention. In the case of the pricing strategy presented, the end-user would have control over a bandwidth management agent which would know the user's preferences with regard to cost minimization, delay control, and bandwidth needs. This agent would then take responsibility for network management at the end-point, in much the same way as applications that utilize RTP [10] react to network congestion issues without the intervention of the user.

#### 6.4 *Charging Receivers*

It has been argued that the vast majority of Internet traffic, primarily due to web-based applications, is initiated by the receiver; therefore, it is only appropriate that the receiver, rather than the sender, pay for the transmission [23]. Our pricing strategy does not preclude receiver payments except to push transactions for such payments to higher layers of the protocol, as is already the case in the current Internet. It is possible to structure the actual payment and accounting system in a variety of ways to accommodate payment by receivers. For example, web servers currently pay for the volume of transmitted traffic by transmitting advertisements to the receiver, or through a subscription service. As already discussed in the previous paragraph, delivery of web content through best-effort service will continue to preserve current modes of payment by the web hosts. Only the delivery of web content at higher levels of quality of service, in conjunction with a pricing framework such as the one presented in this paper, may involve assessing higher subscription fees to the receiver.

#### 6.5 *Conclusion*

In this paper, we have presented a novel pricing strategy based on separating the pricing components due to network resources and due to the preferential service received by a packet. This strategy allows a simple, practical and computationally efficient means to achieve QoS provisioning for multiple classes of service in a DiffServ environment, while also maintaining stable transmission rates from the end users. In particular, our strategy of adding a separate price component for preferential service allows a more efficient market in which the price charged for each service class is determined by the cost of providing such service and the dynamically changing demand for it. This serves to enforce efficiency in capacity management among the various service classes. We have used both queueing-theoretic analysis and simulation to demonstrate the sta-

bility and feasibility of such a pricing strategy. Our pricing strategy can be adapted in a variety of frameworks that achieve congestion control and differentiated services through pricing.

## A Appendix: Pricing Function Derivation

### A.1 Generalized Stability Issues

The system is modeled as an  $M/M/1$  queue with multiple users continually adjusting their own transmission rates in order to optimize the total consumer surplus gained from the network. In order to remove the stability issues associated with the price estimation functions utilized by individual users, we assume that each user receives instantaneous price feedback from the network.

Under this framework, the total amount of traffic submitted to the network by the users can be expressed as:

$$\lambda(t) = \sum_i \min \left\{ \frac{w_i}{P(n(t))}, \lambda_i^{max} \right\} \quad (\text{A.1})$$

Based upon Equation (A.1), the state transition probabilities of the queue are given by:

$$\pi_{n-1,n} = \frac{\sum_i w_i}{P(n)} \quad (\text{A.2})$$

$$\pi_{n,n-1} = \mu \quad (\text{A.3})$$

As stated in [13], ergodic behavior of the Markov chain can be attained under the following conditions:

$$\sum_{n=0}^{\infty} \prod_{k=0}^{n-1} \frac{\lambda_k}{\mu_{k+1}} < \infty \quad (\text{A.4})$$

$$\sum_{n=0}^{\infty} \frac{1}{\lambda_n \prod_{k=0}^{n-1} \frac{\lambda_k}{\mu_{k+1}}} = \infty \quad (\text{A.5})$$

The above conditions can be met given that the following statement is achieved:



$$\frac{\lambda_n}{\mu_{n-1}} < 1 : n \geq n_0 \quad (\text{A.6})$$

We can then also state the following about the Markov state probabilities:

$$p_n = \frac{\left(\frac{\sum_i w_i}{\mu}\right)^n \prod_{i=1}^n \frac{1}{P(i)}}{1 + \sum_{k=1}^{\infty} \left(\frac{\sum_i w_i}{\mu}\right)^k \prod_{i=1}^k \frac{1}{P(i)}} \quad (\text{A.7})$$

## A.2 Polynomial Pricing Function

Under the polynomial pricing strategy presented in [8, 21], the price stamped by the network and the total amount of traffic submitted to the network becomes the following:

$$P(n(t)) = \left(\frac{an(t)}{C} + b\right)^k \quad (\text{A.8})$$

$$\lambda(t) = \sum_i \min \left\{ \frac{w_i}{\left(\frac{a(n(t)+1)}{C}\right)^k}, \lambda_i^{max} \right\} \quad (\text{A.9})$$

$C$  represents the capacity of the buffer.  $n(t)$  is the number of packets currently in the queue.  $b$  represents an overstatement of the current queue state, which is in effect, a base access charge. For simplicity in presentation, we assume that  $b$  is  $\frac{a}{C}$ . This is a minor modification that is tantamount to establishing a minimum access charge for network resources. The resultant Markov state transitions are shown below.

$$\pi_{n-1,n} = \left(\frac{C}{a}\right)^k \frac{\sum_i w_i}{n^k} \quad (\text{A.10})$$

$$\pi_{n,n-1} = \mu \quad (\text{A.11})$$

We can then say the following about the state probabilities:

$$p_n = p_0 (\mathcal{F}(a, k))^n \frac{1}{n!^k} \quad (\text{A.12})$$

$$p_0 = \frac{1}{1 + \sum_{n=1}^{\infty} (\mathcal{F}(a, k))^n \frac{1}{n!^k}} \quad (\text{A.13})$$

where

$$\mathcal{F}(a, k) = \left(\frac{C}{a}\right)^k \frac{\sum_i w_i}{\mu} \quad (\text{A.14})$$

For integer values of  $k > 1$ , the state probabilities and expected buffer occupancy reduces to a function of the generalized hyper-geometric series [12], as shown below:

$$p_0 = \frac{1}{1 + \mathcal{F}(a, k)_1 F_k \left[ \begin{matrix} 1 \\ 2_1, \dots, 2_k \end{matrix}; \mathcal{F}(a, k) \right]} \quad (\text{A.15})$$

$$p_n = \frac{(\mathcal{F}(a, k))^n \frac{1}{n!^k}}{1 + \mathcal{F}(a, k)_1 F_k \left[ \begin{matrix} 1 \\ 2_1, \dots, 2_k \end{matrix}; \mathcal{F}(a, k) \right]} \quad (\text{A.16})$$

$$E[n(t)] = \frac{\mathcal{F}(a, k)_0 F_{k-1} \left[ \begin{matrix} 1 \\ 2_1, \dots, 2_{k-1} \end{matrix}; \mathcal{F}(a, k) \right]}{1 + \mathcal{F}(a, k)_1 F_k \left[ \begin{matrix} 1 \\ 2_1, \dots, 2_k \end{matrix}; \mathcal{F}(a, k) \right]} \quad (\text{A.17})$$

A situation similar to the linear case,  $k = 1$ , is examined in [13]. The buffer distribution is given by:

$$p_0 = e^{-\mathcal{F}(a, k)} \quad (\text{A.18})$$

$$p_n = e^{-\mathcal{F}(a, k)} (\mathcal{F}(a, k))^n \frac{1}{n!} \quad (\text{A.19})$$

$$E[n(t)] = \mathcal{F}(a, k) \quad (\text{A.20})$$

A finite value of the expected buffer occupancy, as discussed in [13], proves stable operation.

## References

- [1] J. Altmann, H. Daanen, H. Oliver, and A. Sánchez-Beato Suárez. How to Market-Manage a QoS Network. In *Proceedings of IEEE INFOCOM*, New York, NY, June 2002.
- [2] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin. REM: Active Queue Management. *IEEE Network*, 15(3):48–53, May–June 2001.
- [3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475: An Architecture for Differentiated Services, December 1998.

- [4] R. Braden, D. Clark, and S. Shenker. RFC 1633: Integrated Services in the Internet Architecture: an Overview, June 1994.
- [5] X.-R. Cao, H.-X. Shen, R. Milioto, and P. Wirth. Internet Pricing with a Game Theoretical Approach: Concepts and Examples. *IEEE/ACM Transactions on Networking*, 10(2):208–216, April 2002.
- [6] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang. Pricing in Computer Networks: Motivation, Formulation, and Example. *IEEE/ACM Transactions on Networking*, 1(6):614–627, December 1993.
- [7] E. W. Fulp and D. S. Reeves. Optimal Provisioning and Pricing of Differentiated Services using QoS Class Promotion. In *Proceedings of the INFORMATIK: Workshop on Advanced Internet Charging and QoS Technology*, Vienna, Austria, September 2001.
- [8] A. Ganesh, K. Laevens, and R. Steinberg. Congestion Pricing and User Adaptation. In *Proceedings of IEEE INFOCOM*, pages 959–965, Anchorage, AK, April 2001.
- [9] R. J. Gibbens and F. P. Kelly. Resource Pricing and the Evolution of Congestion Control. *Automatica*, 35(12):1969–1985, 1999.
- [10] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: RTP: A transport protocol for real-time applications, January 1996.
- [11] A. Gupta, D. O. Stahl, and A. B. Whinston. Priority Pricing of Integrated Services Networks. In L. W. McKnight and J. P. Bailey, editors, *Internet Economics*, pages 323–352. MIT Press, 1997.
- [12] G. H. Hardy. Hypergeometric series. In *Ramanujan: Twelve Lectures on Subjects Suggested by His Life and Work*, chapter 7, pages 101–112. Chelsea, 3rd edition, 1999.
- [13] L. Kleinrock. *Queueing Systems*, volume I: Theory, chapter Birth-Death Queueing Systems in Equilibrium, pages 89–114. Wiley-Interscience, New York, NY, 1975.
- [14] J. K. Mackie-Mason, L. Murphy, and J. Murphy. Responsive Pricing in the Internet. In L. W. McKnight and J. P. Bailey, editors, *Internet Economics*, pages 279–303. MIT Press, Cambridge, MA, 1997.
- [15] J. K. Mackie-Mason and H. R. Varian. Pricing the Internet. In B. Kahin and J. Keller, editors, *Public Access to the Internet*. Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [16] J. K. MacKie-Mason and H. R. Varian. Economic FAQs about the Internet. In L. W. McKnight and J. P. Bailey, editors, *Internet Economics*, pages 27–62. MIT Press, Cambridge, MA, 1997.
- [17] P. Marbach. Pricing Priority Classes in a Differentiated Services Network. In *Proceedings of the 37th Annual Allerton Conference on Communications, Control, and Computing*, Monticello, IL, 1999.

- [18] J. Martin and A. Nilsson. On Service Level Agreements for IP Networks. In *Proceedings of IEEE INFOCOM*, New York, NY, June 2002.
- [19] J. Murphy and L. Murphy. Bandwidth Allocation by Pricing in ATM Networks. In *IFIP Transactions C-24: Broadband Communications II*, pages 333–351, Paris, France, March 1994.
- [20] A. Odlyzko. Paris Metro Pricing for the Internet. In *ACM Conference on Electronic Commerce*, pages 140–147, Denver, CO, November 1999.
- [21] A. J. O’Donnell and H. Sethu. A Novel, Practical Pricing Strategy for Congestion Control and Differentiated Services. In *Proceedings of the IEEE International Conference on Communications*, volume 2, pages 986–990, New York, NY, April–May 2002.
- [22] I. Ch. Paschalidis and J. N. Tsitsiklis. Congestion-Dependent Pricing of Network Services. *IEEE/ACM Transactions on Networking*, 8(2):171–184, April 2000.
- [23] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in Computer Networks: Reshaping the Research Agenda. *ACM Computer Communication Review*, 26(2):19–43, April 1996.
- [24] Y. Tamir and G. L. Frazier. Dynamically Allocated Multi-Queue Buffers for VLSI Communication Switches. *IEEE Transactions on Computers*, 41(6):725–737, June 1992.
- [25] X. Wang and H. Schulzrinne. Pricing Network Resources for Adaptive Applications in a Differentiated Services Network. In *Proceedings of IEEE INFOCOM*, pages 943–952, Anchorage, AK, April 2001.
- [26] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7(5):8–18, September 1993.

# Fair and Efficient Packet Scheduling Using Elastic Round Robin

Salil S. Kanhere, *Student Member, IEEE*, Harish Sethu, *Member, IEEE*, and Alpa B. Parekh

**Abstract**—Parallel systems are increasingly being used in multiuser environments with the interconnection network shared by several users at the same time. Fairness is an intuitively desirable property in the allocation of bandwidth available on a link among traffic flows of different users that share the link. Strict fairness in traffic scheduling can improve the isolation between users, offer a more predictable performance and improve performance by eliminating some bottlenecks. This paper presents a simple, fair, efficient, and easily implementable scheduling discipline, called *Elastic Round Robin (ERR)*, designed to satisfy the unique needs of wormhole switching, which is popular in interconnection networks of parallel systems. In spite of the constraints of wormhole switching imposed on the design, ERR is also suitable for use in Internet routers and has better fairness and performance characteristics than previously known scheduling algorithms of comparable efficiency, including Deficit Round Robin and Surplus Round Robin. In this paper, we prove that ERR is efficient, with a per-packet work complexity of  $O(1)$ . We analytically derive the relative fairness bound of ERR, a popular metric used to measure fairness. We also derive the bound on the start-up latency experienced by a new flow that arrives at an ERR scheduler. Finally, this paper presents simulation results comparing the fairness and performance characteristics of ERR with other scheduling disciplines of comparable efficiency.

**Index Terms**—Fair queuing, Deficit Round Robin, Surplus Round Robin, relative fairness bound, quality of service, wormhole networks.

## 1 INTRODUCTION

IN interconnection networks of parallel systems, packets belonging to different traffic flows often share links in their respective paths toward their destinations. Fairness is an intuitively desirable property in the allocation of bandwidth available on a link among multiple traffic flows that share the link. Fairness in packet scheduling becomes especially desirable with the increasing use of parallel systems in multiuser environments with the interconnection network shared by several users at the same time. Fair allocation of bandwidth at links within a network is a necessary requirement for providing protection to flows, i.e., for ensuring that the performance is not affected when another possibly misbehaving flow tries to send packets at a rate faster than its fair share. In multiuser environments, the protection guaranteed by fair scheduling of packets improves the isolation between users, a quality strongly desired by customers of parallel systems [1]. Isolation offers a more predictable performance to user applications, which also facilitates repeatability of performance results necessary for reliable benchmarking of systems and applications without taking all the users off the system. Strict fairness is also desirable for good performance since unfair treatment of some traffic flows in the network can easily lead to unnecessary bottlenecks.

Most switch architectures designed for interconnection networks of parallel systems, however, eliminate only the worst kinds of unfairness, such as starvation, where packets belonging to one traffic flow may not be scheduled for an indefinite period of time. In the most commonly implemented scheduling discipline, First-Come-First-Served (FCFS), packets are scheduled in the order of their arrival times. The advantage of this scheme is its simplicity since the scheduler is not even required to distinguish packets by the flows to which they belong. The difficulty with FCFS is that it does not provide adequate protection from a sudden bursty source sending packets at a rate higher than its fair share for brief periods of time. Such a source can significantly increase the mean delay of packets belonging to flows from other sources. An alternate technique is Packet-Based Round Robin (PBRR) in which packets are queued separately, based on the flows to which they belong, and the scheduler transmits one packet from each queue in a round-robin fashion [2]. The PBRR scheduler, however, is not fair since flows sending longer packets can use up an unfairly high fraction of the available bandwidth.

One of the difficulties in designing a fair scheduling algorithm for interconnection networks of parallel systems is the unique restriction on the scheduler imposed by wormhole switching [3], a popular technique used in these networks. Wormhole switching and its variations are widely used in a variety of parallel systems and more recently in system area networks [4], [5], [6], [7], [8], [9]. Wormhole switching is distinguished by the fact that the granularity of flow control in the network can be smaller than a packet. This unit of flow control is called a *flit*. In order to not add to the per-flit overhead, only the head flit (the first flit) of a packet contains information necessary to route the packet through the network. A switch in the

- S.S. Kanhere and H. Sethu are with the Department of Electrical and Computer Engineering, Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104. E-mail: {salil, sethu}@ece.drexel.edu.
- A.B. Parekh is with Lockheed Martin Global Telecommunications, Clarksburg, MD 20871. E-mail: alpa.parekh@lmco.com.

Manuscript received 28 June 2000; accepted 12 July 2001.

For information on obtaining reprints of this article, please send e-mail to: [tpds@computer.org](mailto:tpds@computer.org), and reference IEEECS Log Number 112362.

network reads the information in the head flit and directs it to the next switch or the end-system device in its path. The rest of the flits of the packet follow the path of the head flit.

Consider a packet scheduler at an output link in a wormhole switch serving several queues, each corresponding to a different flow and consisting of flits ready to be forwarded to the output link. We define a *queue* as a logical entity containing a sequence of flits that have to be served in a FIFO order. Note that, depending on the buffering architecture of the switch, a queue may not be the same thing as a buffer since a single buffer can implement multiple logical queues [10], [11]. In wormhole switching, flits from different flows cannot be multiplexed on the same link unless each flit is appropriately marked to distinguish it from flits belonging to other flows. In wormhole networks with virtual channels [12], each flit is marked by the virtual channel it belongs to and, therefore, it is possible to time-multiplex flits from different channels on the same physical link. If each flow can be assigned a separate virtual channel, one may use the Flit-Based Round Robin (FBRR) scheduler which visits the flow queues in round-robin fashion, and transmits one flit from each queue. This scheme is very fair among the flows in terms of the number of flits scheduled from each flow over any interval of time. However, this scheme for achieving fairness is prohibitively expensive since it can only be used if there are as many virtual channels implemented as there are flows (which can be in hundreds or thousands). In addition, by serving packets flit-by-flit, the FBRR scheme uniformly increases the delay of packets in all the flows [13]. A packet-by-packet scheduler, therefore, is more desirable since it can achieve a better average delay.

Wormhole switches typically use a flit-by-flit credit-based flow control protocol and, therefore, downstream congestion can thwart the progress of the packet currently being served for an unpredictable length of time. Since, as explained earlier, it may not always be possible to time-multiplex the transmission of packets from different flows, one cannot always begin forwarding packets from another flow until all flits belonging to the packet currently being served are forwarded. Thus, during the time that a packet is in the middle of its transmission, packets from other flows may be blocked without access to the output link even while there are no flits being transmitted over the link. Therefore, the relevant measure of the use of a resource, in this case the output link, is the length of time a flow occupies the link. In wormhole networks, therefore, fairness should be based on the length of time each flow occupies a link and not on the number of flits sent by each flow over the link. This length of time depends on the downstream congestion, which can be hard to predict without complex feedback mechanisms. In wormhole networks, unlike in Internet routers and many other networks, this length of time cannot be accurately estimated from knowledge of the length of the packet being transmitted. The actual length of time that a packet takes to be dequeued, thus, may not be known until the last flit of the packet is dequeued. A scheduling discipline for wormhole networks, therefore, should be able to make a decision on starting the transmission of a packet without knowledge of the length

of time it will take to transmit the entire packet. In addition, the algorithm also cannot assume an upper bound on this length of time. In other words, the unique requirements of wormhole switching require that a scheduler perform its operations without *any* assumptions on how long it will take to transmit a packet.

In traditional scheduling literature, it is typically assumed that the length of time it takes to transmit a packet is directly proportional to the size of the packet. Therefore, the problem of designing a fair scheduler for wormhole networks is equivalent to the problem of designing a fair scheduler in the traditional sense, but without the scheduler making *any* assumptions on the size of a packet before beginning the transmission. Based on this equivalence, we present our paper as a solution to the latter problem. It should be noted that in many real interconnection networks, as well as in Internet routers, packet headers do carry a field with the packet length in it and, therefore, the problem in such cases is not a lack of knowledge of the packet length. However, when a scheduler uses the size of a packet to make its decisions, it cannot be readily adapted to the unique requirements of wormhole switching.

Over the last decade, a variety of algorithms that seek to achieve fairness in bandwidth allocation have been proposed and implemented in Internet routers [14], [15], [16], [17], [18], [19], [20], [21], [22], [23]. A number of these scheduling disciplines are discussed in Section 2. Unfortunately, most fair scheduling disciplines proposed for Internet routers are either too expensive to implement in high-speed hardware switches because of the work complexity of per-packet processing or cannot be easily adapted to the unique requirements of wormhole networks described above. For example, most timestamp-based schedulers, such as Weighted Fair Queuing [24], have a work complexity of  $O(\log n)$  with respect to the number of flows. On the other hand, more efficient schedulers, such as Deficit Round Robin (DRR) [22] and Surplus Round Robin (SRR) [18], [19], [20] require knowledge of the upper bound on packet lengths to achieve a work complexity of  $O(1)$ , rendering them difficult to adapt to wormhole networks. In this paper, we present *Elastic Round Robin (ERR)*, a new, simple, fair, efficient, and low-latency packet scheduling discipline that can be used in both Internet routers and wormhole switches. Besides being efficient, ERR has better fairness and performance characteristics than previously known scheduling algorithms of comparable efficiency, including Deficit Round Robin and Surplus Round Robin.

Section 3 presents the ERR scheduling algorithm along with the rationale behind it. Section 4 presents analytical results on the efficiency, fairness, and the performance characteristics of ERR. We consider a scheduler efficient if the order of the work complexity of enqueueing and dequeuing a packet, with respect to the number of flows, is  $O(1)$ . We prove that the work complexity of ERR is  $O(1)$ , equal to or better than other scheduling disciplines. We measure fairness using a well-known and widely used metric, known as the relative fairness bound [17]. We prove that the relative fairness bound of ERR is  $3m$ , where  $m$  is the size of the largest packet that *actually* arrives during the execution of ERR. Finally, Section 4 also presents bounds on

the start-up latency of ERR, the delay experienced by the first packet of a new flow. A low start-up latency is critical to obtaining low latencies with control packets. The results in this section present an analytical proof that the ERR algorithm has better fairness properties, as well as better performance characteristics, than other fair scheduling disciplines of comparable efficiency, such as DRR and SRR.

Section 5 presents several simulation results on the fairness and performance characteristics of this algorithm in comparison with other algorithms. Section 6 concludes the paper, with a tabulated summary of the properties of ERR in comparison to other fair scheduling algorithms. The concluding section also includes a brief discussion of other applications of ERR.

## 2 BACKGROUND AND PREVIOUS WORK

Traditionally, a *flow* is defined as a sequence of packets generated by the same source and headed toward the same destination via the same path in the network. It is assumed that packets belonging to different flows are queued separately while they await transmission. A *scheduler* dequeues packets from these queues and forwards them for transmission. A flow is said to be *active* during a period if its queue is nonempty throughout this period. A flow is *inactive* when its queue is empty. Note that, even though we use the above traditional definition of a flow to present our results in this paper, a flow can also be more broadly defined as any distinct sequence of packets queued separately at the scheduler and competing with other sequences of packets for service by the scheduler. For example, in parallel systems, a flow may also be defined as the set of all packets belonging to the same user, with packets of these flows queued at the scheduler accordingly.

A precise definition of fairness is essential before further discussion of fair scheduling of flows. The classic notion of fairness in the allocation of a resource among multiple requesting entities with equal rights to the resource, but unequal demands, is as follows [24]:

- The resource is allocated in order of increasing demand.
- No requesting entity gets a share of the resource larger than its demand.
- Requesting entities with unsatisfied demands get equal shares of the resource.

The Generalized Processor Sharing (GPS) algorithm is an unimplementable but ideal scheduling discipline, which satisfies the above notion of absolute fairness [14], [16]. The GPS scheduler visits each active flow's queue in a round-robin fashion and serves an infinitesimally small amount of data from each queue in such a way that during any finite interval of time, it can visit each queue at least once. Consider a set of  $n$  flows denoted by  $1, 2, \dots, n$  demanding bandwidths  $b_1, b_2, \dots, b_n$  on a link of total bandwidth  $B$ . Without loss of generality, assume  $b_1 \leq b_2 \leq \dots \leq b_n$ . The GPS scheduler first allocates  $B/n$  of the bandwidth to each of the active flows. If this is more than the bandwidth demanded by flow 1, the unused bandwidth,  $B/n - b_1$ , is divided equally among the remaining  $n - 1$  flows. If the

total bandwidth allocated thus far to flow 2 is more than  $b_2$ , the unused excess bandwidth is again divided equally, this time among the remaining  $n - 2$  flows. The allocation process of the GPS scheduler continues in this fashion until each flow has received no more than its demand and, if the demand was not satisfied, no less than any other flow with higher demand.

Over the last decade, a variety of algorithms that seek to achieve fairness in bandwidth allocation as per the above definition have been proposed and implemented in Internet routers. Algorithms, such as *Weighted Fair Queuing (WFQ)* [14], [15], try to emulate the ideal GPS scheduler by time-stamping each arriving packet with a *finish number*, the expected completion time of the packet if it were scheduled by the GPS scheduler. The WFQ scheduler then serves the packets in the increasing order of the finish numbers. WFQ is not very efficient in the method it uses to compute the timestamps. In addition, WFQ suffers from the cost associated with sorting among the timestamps, incurring a work complexity of  $O(\log n)$ , where  $n$  is the number of flows. Another scheduling discipline, *Worst-case Fair Weighted Fair Queuing (WF<sup>2</sup>Q)* [21], improves upon the emulation of GPS and makes the implementation easier. In fact, it can be shown that no packet-by-packet scheduler can be more fair than WF<sup>2</sup>Q. However, WF<sup>2</sup>Q suffers from the same work complexity as WFQ.

A simpler implementation is achieved by *Self-Clocked Fair Queuing (SCFQ)* [17], which uses the finish number of the packet currently being transmitted in the computation of finish numbers for arriving packets. SCFQ is slightly less fair than WFQ and also has a larger delay bound. *Start-time Fair Queuing (SFQ)* [23], is a variant of SCFQ which uses the starting time of the packet currently in service to compute the timestamp of the arriving packet. SFQ has better fairness properties than SCFQ and lower worst-case delays. Flows, instead of packets, are timestamped in another approach of similar complexity, called *Time-Shift Scheduling* [25]. In this scheme, the scheduler's real-time clock is periodically adjusted to achieve the effect of bounding the difference between any pair of flow timestamps, thus ensuring fair scheduling of flows. The complexity of timestamp computations is further reduced in two recently proposed timestamp-based algorithms, *Frame-Based Fair Queuing (FFQ)* and *Starting Potential-Based Fair Queuing (SPFQ)* [26]. FFQ, however, uses a framing approach in which the fairness depends on the frame size chosen in the implementation. SPFQ has better fairness properties at a cost of more complexity than FFQ.

None of the scheduling disciplines described above, however, avoid the  $O(\log n)$  work complexity associated with sorting among the timestamps. *Deficit Round Robin (DRR)* [22], a less fair but more efficient scheduling discipline with an  $O(1)$  per-packet work complexity, was proposed by Shreedhar and Varghese in 1996. DRR is not a timestamp-based algorithm and, therefore, avoids the associated computational complexity. DRR serves active flows in a strict round-robin order and succeeds in eliminating the unfairness of pure packet-based round-robin by maintaining a *deficit counter (DC)* to keep an account of past unfairness. A *quantum* is assigned to each of the flows

and when a flow is picked for service, its DC is incremented by the quantum value for that flow. A packet is served from a flow only if the packet size at the head of the flow queue is less than the sum of DC and the quantum value; otherwise, the scheduler begins serving the next flow in the round-robin sequence. When a packet is transmitted, the DC corresponding to that flow is decremented by the size of the transmitted packet.

In DRR, in order that the per-packet work complexity is  $O(1)$ , one has to make sure that the quantum value chosen is no smaller than the size of the largest packet that may potentially arrive at the scheduler [22]. Otherwise the per-packet work complexity increases to  $O(n)$  since one may encounter a situation in which, even after visiting each of the  $n$  flows and examining the respective DC values, no packet is eligible for transmission. A per-packet work complexity of  $O(1)$  is ensured if we make sure that at least one packet is transmitted from each active flow during each round. This is ensured if the quantum is no smaller than the size of the largest possible packet since this guarantees that the packet size at the head of each queue at the start of its service opportunity will always be less than the sum of the DC value and the quantum value of the flow. In order to achieve a per-packet work-complexity of  $O(1)$ , therefore, the DRR scheduler requires knowledge of the upper bound on the size of a packet. Thus, DRR is not ideally suitable for wormhole networks since it requires the knowledge of the size of a packet before making a decision on transmitting it and, in addition requires an upper bound on the size of a packet.

In [18], [19], a fair scheduler similar to DRR was proposed. This algorithm, later known as *Surplus Round Robin (SRR)*, has also been used in other contexts, such as in [20]. SRR is a modified version of DRR in which the scheduler continues serving a flow as long as the DC value of the flow is positive. When the DC becomes negative, the scheduler begins serving the next flow in the round-robin sequence. Thus, while DRR never allows a flow to overdraw its account but rewards an under-served flow in the next round, SRR allows a flow to overdraw its account but penalizes the flow accordingly in the next round. DRR keeps an account of each flow's deficit in service, while SRR keeps an account of the surplus service received by each flow. SRR does not require the scheduler to know the length of a packet before scheduling it. However, it does require the use of a fixed quantum assigned to each flow per round. As in DRR, in order to ensure an  $O(1)$  per-packet work complexity, the quantum value has to be no smaller than the size of the largest packet that may potentially arrive at the scheduler. SRR, like DRR, cannot be readily adapted for use in wormhole switching since it also requires knowledge of the upper bound on packet sizes.

### 3 ELASTIC ROUND ROBIN

Even though ERR was designed for wormhole networks, it can be used in a wide variety of contexts whenever there is a shared resource that needs to be allocated fairly among multiple requesting entities. In some of these contexts, its unique properties relevant to wormhole switching are critical and, in some others, its advantages derive from its

```

Initialize: (Invoked when the scheduler is initialized)
RoundRobinVisitCount = 0;
PreviousMaxSC = 0;
for ( $i = 0$ ;  $i < n$ ;  $i = i + 1$ )
     $SC_i = 0$ ;

Enqueue: (Invoked when a packet arrives)
 $i = \text{QueueInWhichPacketArrives}$ ;
if ( $\text{ExistsInActiveList}(i) == \text{FALSE}$ ) then
    AddToActiveList( $i$ );
    Increment  $\text{SizeOfActiveList}$ ;
     $SC_i = 0$ ;
end if;

Dequeue:
while (TRUE) do
    if ( $\text{RoundRobinVisitCount} == 0$ ) then
        PreviousMaxSC = MaxSC;
        RoundRobinVisitCount =  $\text{SizeOfActiveList}$ ;
        MaxSC = 0;
    end if;
     $i = \text{HeadOfActiveList}$ ;
    RemoveHeadOfActiveList;
     $A_i = 1 + \text{PreviousMaxSC} - SC_i$ ;
    Sent $_i = 0$ ;
    do
        TransmitPacketFromQueue( $i$ );
        Increase Sent $_i$  by  $\text{LengthInFlitsOfTransmittedPacket}$ ;
    while ( $\text{Sent}_i < A_i$ );
     $SC_i = \text{Sent}_i - A_i$ ;
    if ( $SC_i > \text{MaxSC}$ ) then
        MaxSC =  $SC_i$ ;
    end if;
    if ( $\text{QueueIsEmpty} == \text{FALSE}$ ) then
        AddQueueToActiveList( $i$ );
    else
         $SC_i = 0$ ;
        Decrement  $\text{SizeOfActiveList}$ ;
    end if;
    Decrement RoundRobinVisitCount;
end while;

```

Fig. 1. Pseudo-code for ERR.

simplicity, better fairness and better performance characteristics. Because of the wide applicability of our solution and so that this work may be readily understood and used in a broader variety of contexts, we present this algorithm as a solution to the following abstraction of the problem.

Consider  $n$  flows, each with an associated queue with packets in it. The scheduler dequeues packets from these queues according to a scheduling discipline and forwards them for transmission over an output link. As in traditional scheduling problems, we allow that the length of time it takes to dequeue a packet is proportional to the size of the packet—however, to apply this work to wormhole networks, it is required that the scheduling algorithm not make any assumptions about the length of a packet prior to completely transmitting the packet.

We now proceed to describe the ERR scheduler which meets the above requirement. A pseudo-code implementation of the ERR scheduling algorithm is shown in Fig. 1, consisting of *Initialize*, *Enqueue*, and *Dequeue* routines. The *Enqueue* routine is called whenever a new packet arrives at a flow. The *Dequeue* routine is the heart of the algorithm which schedules packets from the queues corresponding to different flows. In this paper, we use a flit as the smallest



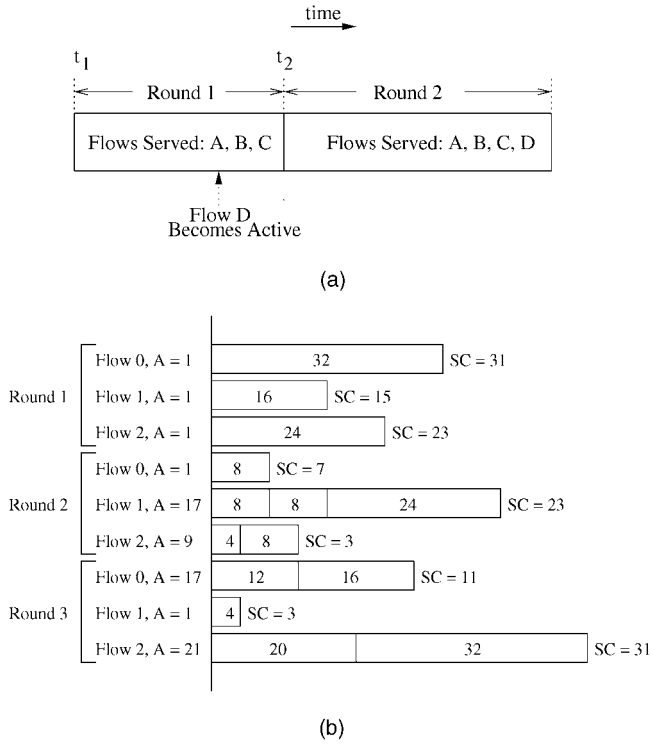


Fig. 2. (a) Definition of a Round and (b) an illustration of three rounds in an ERR execution.

piece of a packet that can be independently scheduled and we measure the length of a packet in flits.

We maintain a linked list, called the *ActiveList*, of flows which are active. A flow whose queue was previously empty and therefore not in the *ActiveList* is added to the tail of the list whenever a new packet belonging to the flow arrives. The ERR scheduler serves the flow  $i$  at the head of this list. After serving flow  $i$ , if the queue of flow  $i$  becomes empty, it is removed from the list. On the other hand, if the queue of flow  $i$  is not empty after it has received its round-robin service opportunity, flow  $i$  is added back to the tail end of the list.

Consider the instant of time,  $t_1$ , when the scheduler is first initialized. We define *Round 1* as one round-robin iteration starting at time  $t_1$  and consisting of visits to all the flows that were in the *ActiveList* at time  $t_1$ . We illustrate this definition of a round using Fig. 2a. Assume that flows  $A$ ,  $B$ , and  $C$  are the only flows active at the beginning of *Round 1*. The visits of the scheduler to flows  $A$ ,  $B$ , and  $C$ , comprise *Round 1*. Let flow  $D$  become active after the time instant  $t_1$ , but before the completion of *Round 1*. Let the time instant  $t_2$  mark the completion of *Round 1*. The scheduler does not visit flow  $D$  in *Round 1* since  $D$  was not in the *ActiveList* at the start of *Round 1*. *Round 2* is now defined as consisting of the visits to all of the flows that are in the *ActiveList* at time  $t_2$ . Assuming that flows  $A$ ,  $B$ , and  $C$  are still active at time  $t_2$ , *Round 2* will consist of visits to the flows  $A$ ,  $B$ ,  $C$ , and  $D$ . In general, we define round  $i$  recursively as the set of visits to all the flows in the *ActiveList* at the instant round  $(i - 1)$  is completed. In order that the scheduler knows the number of flows it has to visit in any given round, we introduce the quantity *RoundRobinVisitCount* which denotes the number of flows that are in the *ActiveList* at the start of a round.

*RoundRobinVisitCount* is decremented by one after each flow is served and, when it eventually equals zero, it implies the end of a round.

In each round, the scheduling algorithm determines the number of flits that a flow is allowed to send. We call this quantity the *allowance* for the flow during that round. The allowance assigned to flow  $i$  during round  $r$  is denoted by  $A_i(r)$ . This allowance, however, is not a rigid one and is actually *elastic* in that a flow may be allowed to send more flits in a round than its allowance. Let  $Sent_i(r)$  be the number of flits that are transmitted from the queue of flow  $i$  in round  $r$ . The ERR scheduler will begin serving the next packet from the queue if the total number of flits transmitted by the flow so far in the current round is less than its allowance. The ERR scheduler, thus, makes the scheduling decision without any knowledge about the packet length. Note that the last packet transmitted by a flow may cause it to exceed its allowance, as can happen when the allowance is smaller than the size of the packet at the head of the corresponding queue. When a flow ends up sending more than its allowance, it is interpreted as having obtained more than its fair share of the bandwidth. The scheduler records this unfairness in the *Surplus Count* (SC) associated with each flow. The surplus count, during any round, is the number of flits the flow sent in addition to its allowance. Let  $SC_i(r)$  denote the surplus count of flow  $i$  in round  $r$ . Then, after serving flow  $i$  in round  $r$ , the scheduler computes  $SC_i(r)$  as follows:

$$SC_i(r) = Sent_i(r) - A_i(r). \quad (1)$$

Let  $MaxSC(r)$  denote the largest surplus count among all the flows served during round  $r$ . This quantity is used to recursively compute the allowances for each of the flows in the next round, using the following equation:

$$A_i(r) = 1 + MaxSC(r - 1) - SC_i(r - 1). \quad (2)$$

Note that, for the flow with the largest surplus count in the previous round, the new allowance is 1. This is ensured by the addition of 1 in (2) so that the scheduler will transmit at least one packet from this flow during the next round.

The allowance given to each of the flows in a given round is not fixed and is computed depending on the behavior of the flows in the previous round. After the ERR scheduler serves flow  $i$ , if the queue of flow  $i$  is empty, its surplus count is reset to zero and it is removed from the *ActiveList*. Otherwise, if flow  $i$  has packets in its queue that are ready for transmission, it is added back at the tail end of the list.

Fig. 2b illustrates the first three rounds in an execution of the ERR scheduling discipline. In this figure, at the beginning of the first of these rounds, the surplus counts for all three flows and the *MaxSC* are all initialized to 0. Thus, from (2), the allowance during round 1 is equal to 1 for all the flows. The sizes of the packets actually sent by the flow during this round are shown by the horizontal bars and the new allowances for the next round are again computed using (1) and (2). It is easily observed from the figure that, in general, flows which receive very little service in a round are given an opportunity to receive proportionately more service in the next round.

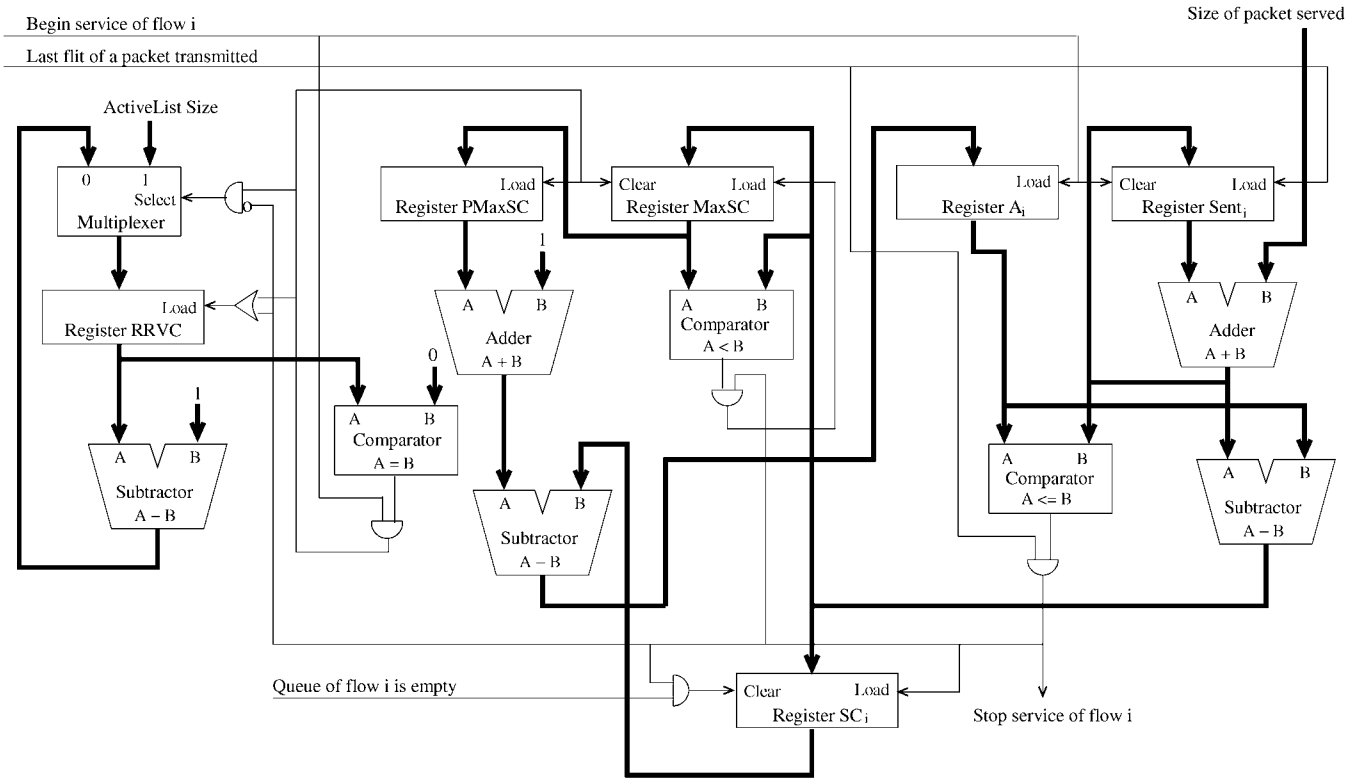


Fig. 3. A block diagram illustration of ERR.

Fig. 3 shows a block diagram of a portion of ERR to illustrate the various operations used to determine when the scheduler should stop service of one flow and begin service for another. In the diagram, *PreviousMaxSC* and *RoundRobinVisitCount* are abbreviated as *PMaxSC* and *RRVC*. The thin lines in the figure indicate single-bit signals, while the thick dark lines indicate multibit buses carrying quantities, such as packet sizes and values of various counters.

**Weighted ERR.** In the interests of clarity of presentation, in this section, we have discussed the ERR scheduler assuming that all the traffic flows have been given equal weights, i.e., equal rights to the bandwidth of the output link. One may, however, want some flows to receive a greater share of the bandwidth than some other flows. Let the weight associated with flow  $i$  be  $w_i$ , indicating its relative share of the bandwidth. The computation of the surplus count for the weighted ERR scheduler is identical to that of the unweighted version. However the allowance calculated by the Weighted ERR scheduler uses the following modified version of (2):

$$A_i(r) = w_i(1 + \text{MaxSC}(r-1)) - \text{SC}_i(r-1). \quad (3)$$

## 4 ANALYTICAL RESULTS

In this section, we analytically derive the work complexity, fairness, and delay properties of ERR.

### 4.1 Work Complexity

Consider an execution of the ERR scheduling discipline over  $n$  flows. We define the work complexity of the ERR scheduler as the order of the time complexity with respect to  $n$  of enqueueing and then dequeuing a packet for transmission.

**Theorem 1.** *The work complexity of an ERR scheduler is  $O(1)$ .*

**Proof.** We prove the theorem by showing that enqueueing and dequeuing a packet are each of time complexity  $O(1)$ .

The time complexity of enqueueing a packet is the same as the time complexity of the *Enqueue* procedure in Fig. 1, which is executed whenever a new packet arrives at a flow. Determining the flow at which the packet arrives is an  $O(1)$  operation. The flow at which the new packet arrives is added to the *ActiveList* if it is not already in the list. This addition of an item to the tail of a linked list data structure is also an  $O(1)$  operation.

We now consider the time complexity of dequeuing a packet. During each service opportunity, the ERR scheduler transmits at least one packet. Thus, the time complexity of dequeuing a packet is equal to or less than the time complexity of all the operations performed during each service opportunity. Each execution of the set of operations inside the while loop of the *Dequeue* procedure in Fig. 1, represents all operations performed during each service opportunity given to a flow. These operations include determining the next flow to be served, removing this flow from the head of the *ActiveList* and possibly adding it back at the tail. All of these operations on a linked list data structure can be executed in  $O(1)$  time. Additionally, each service opportunity includes updating the values of surplus count and allowance corresponding to the flow being served and updating the values of *MaxSC*, *PreviousMaxSC*, *SizeOfActiveList* and *RoundRobinVisitCount*. All of these can be done in constant time, as represented by the constant number of operations in the dequeue procedure in Fig. 1.  $\square$

## 4.2 Fairness

The fairness of a scheduling discipline is best measured in comparison to the GPS scheduling algorithm. This quantity, known as the *Absolute Fairness Bound* of a scheduler  $S$ , is defined as the upper bound on the difference between service received by a flow under  $S$  and that under GPS over all possible intervals of time. This bound is often difficult to derive analytically and, therefore, another fairness metric first proposed in [17] is more commonly employed. This metric, known as the *Relative Fairness Bound (RFB)*, is defined as the maximum difference in the service received by any two flows over all possible intervals of time. The following provides a more rigorous definition. In the following, a flow is considered *active* during an interval of time if, during this interval, its queue is never empty of packets awaiting transmission.

**Definition 1.** Let  $Sent_i(t_1, t_2)$  be the number of flits transmitted by flow  $i$  during the time interval between  $t_1$  and  $t_2$ . Given an interval  $(t_1, t_2)$ , we define the Relative Fairness,  $RF(t_1, t_2)$  for this interval as the maximum value of  $|Sent_i(t_1, t_2) - Sent_j(t_1, t_2)|$  over all pairs of flows  $i$  and  $j$  that are active during this interval. Define the relative fairness bound (RFB) as the maximum of  $RF(t_1, t_2)$  over all possible time intervals  $(t_1, t_2)$ .

**Definition 2.** Define  $m$  as the size in flits of the largest packet that is actually served during the execution of a scheduling algorithm.

**Definition 3.** Define  $M$  as the size in flits of the largest packet that may potentially arrive during the execution of a scheduling algorithm. Note that  $M \geq m$ .

**Lemma 1.** For any flow  $i$  and round  $r$  in the execution of an ERR scheduling discipline,  $0 \leq SC_i(r) \leq m - 1$ .

**Proof.** The lower bound on  $SC_i(r)$  in the expression of the lemma is obvious since the ERR algorithm always schedules at least as many flits as  $A_i(r)$  during round  $r$ . The only exception is when the queue for flow  $i$  becomes empty in round  $r$ , in which case the surplus count of the flow is reset to 0.

The ERR algorithm never begins dequeuing a new packet in a flow after the number of flits sent in a round  $r$  is equal to or more than the allowance  $A_i(r)$ . Thus, the lowest value of  $(Sent_i(r) - A_i(r))$  at which a new packet transmission may begin is 1 and this will be the last packet transmitted by the flow during this round. Since the size of this packet can be no greater than  $m$ , from (1), the upper bound in the expression of the lemma is proven.  $\square$

The following corollary follows directly from Lemma 1.

**Corollary 1.** In any round  $r$ ,  $0 \leq MaxSC(r) \leq m - 1$ .

**Theorem 2.** Given  $n$  consecutive rounds starting from round  $k$ , during which flow  $i$  is active, the bounds on the total number of flits,  $N$ , transmitted by flow  $i$  are given by

$$n + \sum_{r=k-1}^{k+n-2} MaxSC(r) - (m-1) \leq N \leq n + \sum_{r=k-1}^{k+n-2} MaxSC(r) + (m-1).$$

**Proof.** Substituting for  $A_i(r)$  using (2) into (1), we get,

$$Sent_i(r) = 1 + MaxSC(r-1) - SC_i(r-1) + SC_i(r). \quad (4)$$

Summing the LHS in (4) above for  $r = k$  to  $r = k + n - 1$ , we get  $N$ , the total number of flits sent during the  $n$  consecutive rounds under consideration. Equating this to the summation of the RHS in (4) for  $r = k$  to  $r = k + n - 1$ ,

$$N = n + \sum_{r=k-1}^{k+n-2} MaxSC(r) + SC_i(k+n-1) - SC_i(k-1). \quad (5)$$

Using Lemma 1,  $0 \leq SC_i(k+n-1) \leq m-1$ , and  $0 \leq SC_i(k-1) \leq m-1$ . The result of the theorem is readily obtained by substituting for these bounds on  $SC_i(k-1)$  and on  $SC_i(k+n-1)$  in (5).  $\square$

We now proceed to prove the bound on the relative fairness of the ERR scheduling discipline. Note that the relative fairness bound, RFB, is defined taking into consideration all possible intervals of time  $(t_1, t_2)$ . In the following, we prove that a tight upper bound can be obtained considering only a subset of all possible time intervals. This subset is the set of all time intervals bounded by time instants that coincide with the beginning or the end of the service opportunity of flows.

**Definition 4.** Let  $\mathbf{T}$  be the set of all time instants during an execution of the ERR algorithm. Define  $\mathbf{T}_s$  as the set of all time instants at which the scheduler ends serving one flow and begins serving another. Define  $F(t)$ , for  $t \notin \mathbf{T}_s$ , as the flow which is being served at time instant  $t$ . For  $t \in \mathbf{T}_s$ , we define  $F(t)$  as the flow just about to begin service.

The following lemma allows us to prove an upper bound on the relative fairness, stated in Theorem 3, considering only the time intervals  $(t_1, t_2)$ , where  $t_1, t_2 \in \mathbf{T}_s$ .

**Lemma 2.**  $RFB = \max_{t_1, t_2 \in \mathbf{T}_s} RF(t_1, t_2)$ .

**Proof.** This lemma is proven if, for any  $t_1, t_2 \in \mathbf{T}$ , we can find  $t'_1, t'_2 \in \mathbf{T}_s$ , such that  $RF(t'_1, t'_2) \geq RF(t_1, t_2)$ .

Consider any two active flows  $i$  and  $j$  during the time interval between  $t_1$  and  $t_2$ , where  $t_1, t_2 \in \mathbf{T}$ . Without loss of generality, assume that, during this time interval, more flits have been scheduled from flow  $i$  than from flow  $j$ . By appropriately choosing  $t'_1$  as the time instant at either the beginning or the end of the service opportunity given to  $F(t_1)$  at time  $t_1$ , one may verify that  $RF(t'_1, t_2) \geq RF(t_1, t_2)$ . Similarly, an appropriate choice of  $t'_2$ , as either the beginning or the ending instant of the service opportunity given to  $F(t_2)$  at time  $t_2$ , can lead to  $RF(t'_1, t'_2) \geq RF(t_1, t_2)$ .  $\square$

**Theorem 3.** For any execution of the ERR scheduling discipline,  $RFB < 3m$ .

**Proof.** By the statement of Lemma 2, we need to only consider all time intervals bounded by time instants that coincide with the starting or ending of service to a flow. We therefore prove the statement of the theorem using the time interval between instants  $t_1$  and  $t_2$ , where both  $t_1$  and  $t_2$  belong to  $\mathbf{T}_s$ .

Consider any two flows  $i$  and  $j$  that are active in the time interval between  $t_1$  and  $t_2$ . From the algorithm in Fig. 1, it follows that after flow  $i$  receives service, it is added to the tail end of the *ActiveList*. The ERR scheduler then visits flow  $j$ , which is served before flow  $i$  receives service again. Thus, in between any two consecutive service opportunities given to flow  $i$ , flow  $j$  receives exactly one service opportunity. Hence, if  $n_i$  and  $n_j$  denote the total round-robin opportunities received by flows  $i$  and  $j$ , respectively, in the time interval  $(t_1, t_2)$ , then  $|n_i - n_j| \leq 1$ .

Let  $r(t)$  denote the round in progress at time instant  $t$ . Also note that the time instant  $t_1$  may be such that the service opportunity received by one of the two flows in round  $r(t_1)$  may not be a part of interval  $(t_1, t_2)$ . Thus, the first time that the scheduler visits this flow in the interval under consideration would be in the round following  $r(t_1)$ . Consequently, if  $r_i$  and  $r_j$  denote the rounds in which flows  $i$  and  $j$  receive service for the first time in the interval  $(t_1, t_2)$ , respectively, then  $|r_i - r_j| \leq 1$ .

Without loss of generality, we can assume that in the interval  $(t_1, t_2)$ , flow  $i$  starts receiving service before flow  $j$ . Thus,

$$r_j \leq r_i + 1 \quad \text{and} \quad n_i \leq n_j + 1. \quad (6)$$

From Theorem 2, for flow  $i$ ,

$$Sent_i(t_1, t_2) \leq n_i + \sum_{k=r_i-1}^{r_i+n_i-2} MaxSC(k) + (m-1). \quad (7)$$

For flow  $j$ ,

$$n_j + \sum_{k=r_j-1}^{r_j+n_j-2} MaxSC(k) - (m-1) \leq Sent_j(t_1, t_2). \quad (8)$$

Combining (7) and (8) and using (6), we get

$$Sent_i(t_1, t_2) - Sent_j(t_1, t_2) \leq 1 + 2(m-1) + \sum_{k=r_i-1}^{r_i+n_i-2} MaxSC(k) - \sum_{k=r_j-1}^{r_j+n_j-2} MaxSC(k). \quad (9)$$

Let us now consider the quantity  $D$  given by

$$D = \sum_{k=r_i-1}^{r_i+n_i-2} MaxSC(k) - \sum_{k=r_j-1}^{r_j+n_j-2} MaxSC(k).$$

We now compute  $D$  for each of the four possible cases.

Case 1 ( $r_i = r_j$ ,  $n_i = n_j$ ):

$$D = 0.$$

Case 2 ( $r_i = r_j$ ,  $n_i = n_j + 1$ ):

$$D = MaxSC(r_i + n_i - 2).$$

Case 3 ( $r_i = r_j - 1$ ,  $n_i = n_j$ ):

$$D = MaxSC(r_i - 1) - MaxSC(r_i + n_i - 1).$$

Case 4 ( $r_i = r_j - 1$ ,  $n_i = n_j + 1$ ):

$$D = MaxSC(r_i - 1).$$

Using Corollary 1, it is readily verified that, in each of the above four cases,  $D < m$ . Substituting in (9), the statement of the theorem is proven.  $\square$

Note that for the Weighted ERR scheduler, the relative fairness over an interval  $(t_1, t_2)$  is defined as

$$RF(t_1, t_2) = \max_{i,j} \left| \frac{Sent_i(t_1, t_2)}{w_i} - \frac{Sent_j(t_1, t_2)}{w_j} \right|, \quad (10)$$

where  $i$  and  $j$  are flows that are active during the interval  $(t_1, t_2)$  [17], [24].

It can be verified that Theorem 3 can also be proven for the Weighted ERR scheduler using (3) in place of (2) in the proof above.

In comparison to a relative fairness bound of  $3m$  for ERR, both DRR and SRR have a relative fairness bound of  $M + 2m$ , where  $M$  is the size of the largest packet that may *potentially* arrive during the lifetime of the execution of the scheduling discipline. Recall that  $m$  is the size of the largest packet that *actually* arrives during the execution of the scheduler. In most networks, including the Internet, the vast majority of the packets in the traffic are of much smaller size than the maximum possible size of a packet [27]. The value of  $m$  in the expression for the relative fairness, especially over short intervals of time, is likely to be much smaller than  $M$ . The fairness achieved by ERR, thus, is always equal to or better than that achieved by DRR or SRR.

### 4.3 Start-Up Latency

The performance of scheduling disciplines can be compared in terms of at least two quantities: The time it takes for the first packet of a flow to be served completely and the average queuing delay of packets in the flows. We denote the first quantity as the *Start-Up Latency Bound* and define it as the maximum length of time between the instant the first packet of a new flow arrives in its queue and the instant the last flit of this packet is scheduled. A good bound on the start-up latency guarantees that the scheduler will serve a new flow or a flow that has just become active within an acceptable and finite amount of time. This bound is particularly relevant to scheduling of control packets, which typically do not arrive as part of a traffic stream and which have a low tolerance for delays. We define the *queuing delay* of a packet as the length of time between the instant it is placed in the queue for scheduling and the instant its last flit is scheduled. In this section, we analyze the start-up latency of ERR in comparison with other scheduling algorithms. A simulation study of queuing delays is presented in Section 5.

As in the rest of this section, we present our analysis assuming equal weights for all the flows. Therefore, given  $n$  active flows, the share of the bandwidth allowed to each flow is  $r/n$ , where  $r$  is the capacity of the output link. The following theorem proves the start-up latency bound for ERR.

**Theorem 4.** During an execution of the ERR scheduling discipline serving  $n$  active flows at a link of maximum rate  $r$ , the start-up latency,  $S_{ERR}$ , of a newly active flow has an upper bound given by

$$S_{ERR} \leq \frac{(2m-1)n + m}{r}.$$

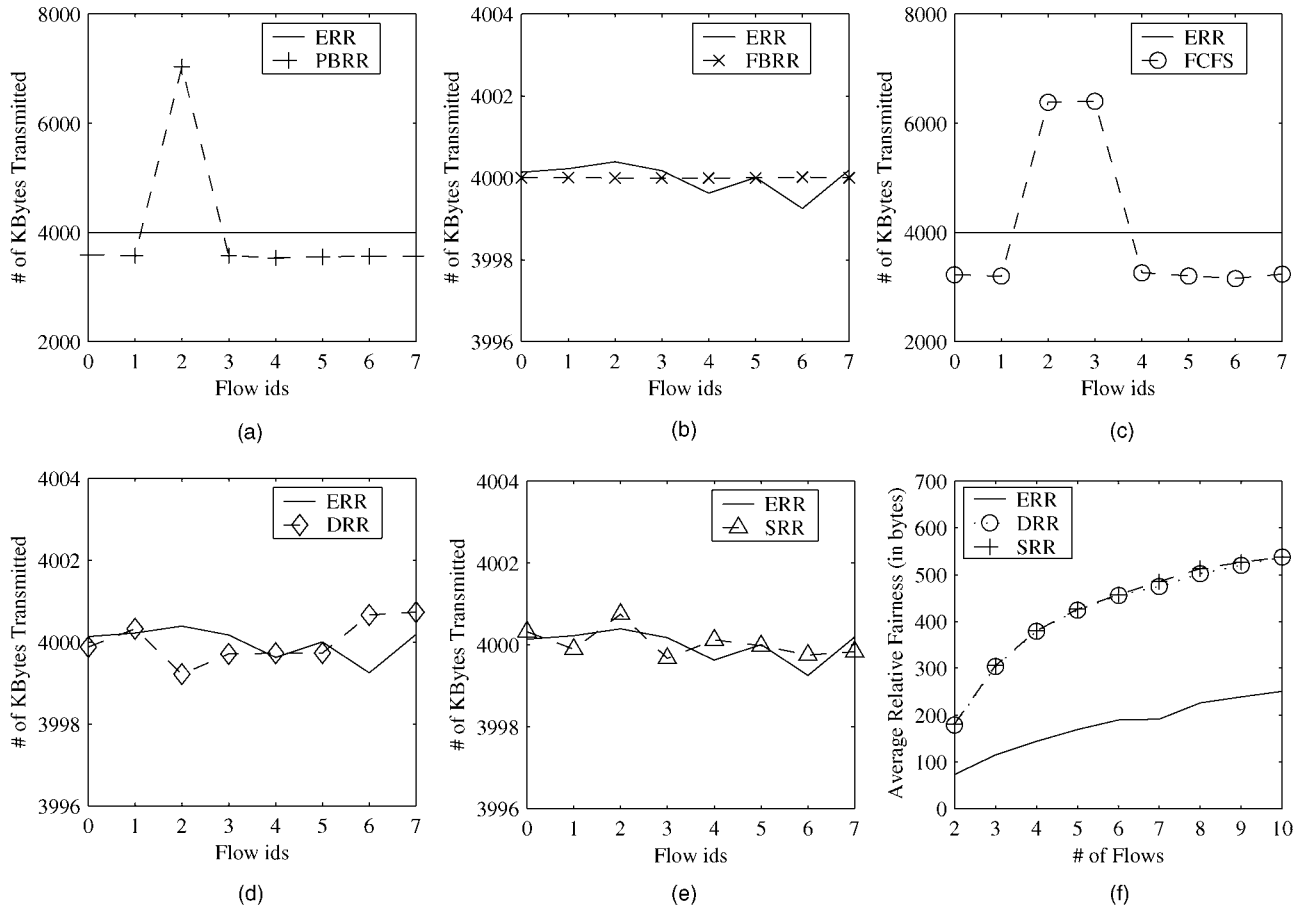


Fig. 4. Simulation results on fairness comparisons.

**Proof.** When the first packet of a newly active flow arrives, the flow is placed at the tail of the *ActiveList* and is served after all the  $n$  previously active flows are served. Using Lemma 1 and Corollary 1 in (4), we get

$$Sent_i(r) \leq 2m - 1.$$

Therefore, a maximum of  $(2m - 1)n$  flits may be served before the scheduler begins serving the newly active flow. Noting that the maximum size of the first packet of the newly active flow is  $m$ , the statement of the theorem is readily proven.  $\square$

The start-up latency bound of ERR is lower than that of DRR, which has a start-up latency,  $S_{DRR}$ , bounded above as follows:

$$S_{DRR} \leq \frac{(M + m - 1)n + m}{r}.$$

It can be readily proven that the start-up latency bound of SRR is equal to that of DRR.

## 5 SIMULATION RESULTS

In this section, we present simulation results on the performance and the fairness properties of the ERR scheduler. The ERR algorithm is compared with other packet-by-packet scheduling algorithms of equivalent work complexity

such as DRR, SRR, FCFS, FBRR, and PBRR. DRR and SRR are the scheduling disciplines that come closest to ERR in terms of being both fair and efficient and, therefore, a majority of our focus is on ERR in comparison with DRR and SRR.

We compare the fairness of scheduling disciplines by plotting for a given interval, during which all the flows are active, the number of bytes scheduled from each of the different flows. Figs. 4a, 4b, 4c, 4d, 4e, and 4f show the results of our simulation experiments on fairness.

For the results in Figs. 4a, 4b, 4c, 4d, and 4e, we simulate eight flows with flow ids from 0 to 7. We collect results for a period of four million cycles, during which we ensure that all the flows are active. The arrival rate in terms of packets per second into the queue corresponding to flow 3 is twice the rate of other flows. Also, the packet lengths are uniformly distributed between one and 64 flits for all the flows, except flow 2. Packets belonging to flow 2 have lengths uniformly distributed between 1 and 128 flits. Note that, in this experiment, the maximum possible packet size,  $M$ , is equal to 128, while the largest packet that actually arrives is also 128 since the number of cycles in the simulation experiment is large. We assume a flit size of 8 bytes and that the scheduler dequeues one flit from one of the queues in each cycle.

Fig. 4a demonstrates that ERR is fair in terms of throughput achieved by the different flows, while PBRR is

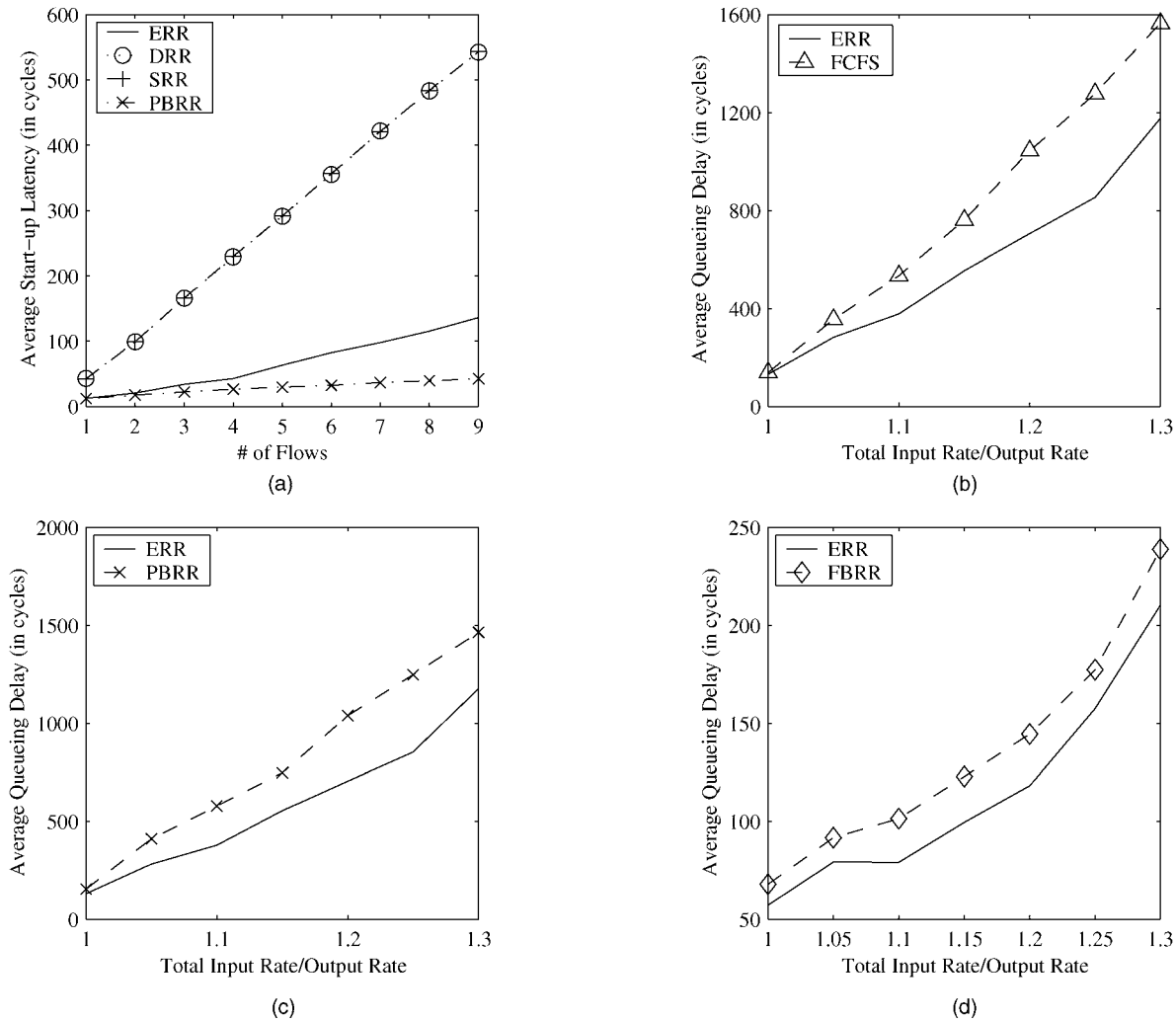


Fig. 5. Simulation results on performance comparisons.

not. As shown in the figure, with PBRR, the flow sending larger packets (flow 2) gains an unfair fraction of the bandwidth.

Fig. 4b shows that ERR, however, is not as fair as FBRR. This is expected since, with a scheduling granularity of one flit, FBRR is the fairest algorithm in terms of throughput achieved by the flows. Note that, as stated by Theorem 3 for ERR, the maximum difference between the number of bytes served from different flows is less than  $3 \times 128 \times 8$  bytes or 3 KBytes.

Fig. 4c compares FCFS with ERR. As expected, in FCFS, flows which send at twice the rate or which send packets of twice the lengths manage to steal approximately twice the bandwidth. ERR, on the other hand, maintains fairness among the flows independent of packet lengths or injection rates.

Fig. 4d compares ERR with DRR and shows that the two scheduling disciplines, for uniformly distributed packet lengths, are comparable in fairness. Fig. 4e similarly shows that ERR and SRR are comparable in fairness.

Fig. 4f shows the result of a simulation in which packet lengths in all the flows are exponentially distributed between one to 64 flits, with the parameter  $\lambda = 0.2$ . Recall

that the relative fairness bound for both DRR and SRR is  $M + 2m$ , whereas that for ERR is  $3m$ . This difference in fairness between the scheduling disciplines is best highlighted by a packet length distribution in which the larger size packets are less likely to appear than smaller size packets, such as when the lengths are exponentially distributed. We compute the average relative fairness achieved by the ERR, DRR, and SRR scheduling disciplines over 10,000 randomly chosen intervals during a period of four million cycles. Fig. 4f demonstrates that when larger size packets are less likely than smaller size packets, which is often the case in many real networks including interconnection networks for parallel systems and the Internet [27], ERR achieves better fairness than DRR or SRR.

Figs. 5a, 5b, 5c, and 5d show the results of our next set of experiments on the performance of the ERR scheduler. In Fig. 5a, we compare the start-up latencies of ERR, DRR, SRR, and PBRR. We do not plot results for FCFS in this figure since, at an FCFS scheduler, the start-up latency is determined by the number of packets that are already waiting in the queue and is thus bounded only by the size

TABLE 1  
Comparison of Fair Scheduling Algorithms

Scheduling Discipline	Complexity	Fairness	Start-up Latency Bound	Applicable to Wormhole Networks
GPS [16]	-	0	$\frac{mn}{r} + \frac{m}{r}$	-
Packet-Based Round Robin [2]	$O(1)$	$\infty$	$\frac{mn}{r} + \frac{m}{r}$	✓
First-Come-First-Served	$O(1)$	$\infty$	$\infty$	✓
Weighted Fair Queueing [14]	$O(\log n)$	$O(n)$	$\frac{mn}{r} + \frac{m}{r}$	-
Self-Clocked Fair Queueing [17]	$O(\log n)$	$2m$	$\frac{2mn}{r} + \frac{m}{r}$	-
Worst-Case Fair Weighted Fair Queueing [21]	$O(\log n)$	$2m$	$\frac{mn}{r} + \frac{m}{r}$	-
Deficit Round Robin [22]	$O(1)$	$M + 2m$	$\frac{(M + m - 1)n}{r} + \frac{m}{r}$	-
Surplus Round Robin [18-20]	$O(1)$	$M + 2m$	$\frac{(M + m - 1)n}{r} + \frac{m}{r}$	-
Elastic Round Robin	$O(1)$	$3m$	$\frac{(2m - 1)n}{r} + \frac{m}{r}$	✓

of the buffers implementing the queues. Recall that the latency bound for ERR is  $\frac{(2m-1)n+m}{r}$ , whereas that for DRR or SRR is  $\frac{(M+m-1)n+m}{r}$ . To best illustrate this difference, we use packet lengths that are exponentially distributed in the range between 1 to 64 flits, with the parameter  $\lambda = 0.2$ . The start-up latency depends on the number of active flows before the arrival of the first packet of the new flow, and therefore, we produce simulation results using different numbers of flows ranging from  $n = 1$  to 9. While these flows are kept active throughout the duration of this simulation, we simulate another flow that alternates between active and inactive periods. The beginning and the end of active periods is chosen randomly. For each active period of the flow, we measure the start-up latency as the number of cycles between the arrival of the first packet of the active period of the flow and the scheduling of the last flit of this packet. For each of the different numbers of flows, Fig. 5a plots the average start-up latency over 1,000 active periods and shows that ERR has better start-up latencies than both DRR and SRR. Note that, even though PBRR has lower start-up latencies, its relative

fairness bound is infinity and cannot be used for achieving fairness.

Figs. 5b and 5c show the average queuing delay of packets arriving at an ERR scheduler in comparison to FCFS and PBRR. In comparing the queuing delays experienced by packets with different scheduling disciplines, it only makes sense to consider flows that are active. If the sum of the rates at which the packets are arriving in the flows is greater than the maximum possible output rate, the delays will eventually reach infinity because of persistent congestion, rendering a meaningful comparison impossible. On the other hand, if there is no congestion at all, queuing delays are all nearly zero. For an appropriate and realistic comparison, therefore, we create transient periods of congestion during which the sum of the input rates is higher than the output bandwidth. In our simulations, these transient periods of congestion last 10,000 cycles, after which we halt all injection of packets into the queues and continue simulation until all the queues are empty. The figures are plotted for the average queuing delay of a packet against the intensity of the transient congestion (measured by the ratio of the sum of the input rates to the maximum possible output rate). We use four flows in the simulations in Figs. 5b, and 5c and, as before, packet arrival rate in the queue for flow 3 is twice that of other flows. Also, as before, the

packet lengths are uniformly distributed from one to 64 flits, except for flow 2, in which the packet lengths are uniformly distributed between one and 128 flits.

Fig. 5b shows that ERR has a better average queuing delay than FCFS when packet sizes and input rates of the flows can be different. A well-known result in queuing theory states that, for any given flow, if a scheduling discipline achieves better average delay than FCFS, it comes at the expense of increasing the delays of some other flows [28]. The better overall average delay of ERR is achieved at the expense of flows sending at twice the rate or flows sending larger packets. FCFS, on the other hand, would have given these flows higher bandwidths and, therefore, lower delays, while increasing the delays of all other flows. Similarly, ERR has a much better average queuing delay than PBRR, as shown in Fig. 5c.

Fig. 5d shows the average queuing delay of packets obtained with ERR and FBRR. Note that in ERR, once a packet begins transmission, the entire packet is completely scheduled before another packet is allowed to begin transmission. This reduces the average delay of packets in comparison to FBRR, which, by serving packets flit-by-flit, uniformly increases the delays experienced by all the flows. In general, for this reason, packet-by-packet round-robin schedulers have the potential to improve average delays in comparison to FBRR. This additional advantage of ERR over FBRR is shown in Fig. 5d.

DRR, SRR, and ERR have finite and close relative fairness bounds and, therefore, the differences between these disciplines in terms of the exact time a packet gets scheduled, is bounded and finite. Therefore, the differences between the average queuing delays with these two disciplines remain small in comparison to the large queuing delays experienced during congestion. Therefore, the queuing delays among ERR, DRR, and SRR are not compared.

## 6 SUMMARY AND CONCLUSION

In this paper, we have presented a novel scheduling discipline called *Elastic Round Robin* (ERR), which is simple, fair, and efficient with a low start-up latency. In addition, it satisfies the unique requirement imposed by wormhole switching on fair scheduling disciplines. We have shown that the work complexity of ERR is  $O(1)$  and, therefore, can be easily implemented in networks with large numbers of flows. In comparison to other scheduling disciplines of similar efficiency, such as Deficit Round Robin (DRR) and Surplus Round Robin (SRR), ERR has better fairness properties, as well as a lower start-up latency bound. Table 1 summarizes the work complexity, fairness, and the start-up latency of several scheduling algorithms. In the column on work complexity,  $n$  is the total number of active flows. In the column on start-up latency,  $n$  is the number of active flows at the instant before the start of the new flow. The peak rate of the output link is denoted by  $r$ . Among scheduling disciplines of comparable efficiency, DRR and SRR come closest to ERR in fairness. However, neither DRR nor SRR is ideally suitable for use in wormhole networks, where the length of time a packet occupies the link is not known before a decision to transmit the packet is

made. On the other hand, ERR can be readily used in wormhole networks, in addition to being perfectly suitable for achieving fair scheduling in Internet routers.

Finally, it is worthwhile to note that ERR can be used in a wide variety of contexts whenever there is a shared resource that needs to be allocated fairly among multiple requesting entities. For example, one may define a flow as the stream of packets belonging to the same virtual channel, in which case, ERR can be used to achieve fairness among virtual channels in the forwarding of flits to the output link, while also achieving lower delays as shown in Fig. 5d. ERR can also be used in the forwarding of packets from the input buffers to the output buffers of switching elements in networks. ERR can also be a solution in token ring networks, where the bandwidth of the ring has to be shared among multiple sources. Similarly, ERR can be used to efficiently arbitrate access to a busy shared bus. The lower start-up latency of ERR among similarly efficient algorithms is especially useful here in improving the latency of short control messages. Finally, ERR is particularly relevant to the problem of job scheduling in operating systems, where multiple processes are competing for limited CPU cycles.

## ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation CAREER Award CCR-9984161 and US Air Force Contract F30602-00-2-0501.

## REFERENCES

- [1] H. Sethu, C.B. Stunkel, and R.F. Stucke, "IBM RS/6000 SP Large System Interconnection Network Topologies," *Proc. Int'l Conf. Parallel Processing*, Aug. 1998.
- [2] J. Nagle, "On Packet Switches with Infinite Storage," *IEEE Trans. Comm.*, vol. 35, no. 4, Apr. 1987.
- [3] W.J. Dally and C.L. Seitz, "The Torus Routing Chip," *J. Distributed Computing*, vol. 1, no. 3, pp. 187-196, Oct. 1986.
- [4] C.B. Stunkel, "The SP2 High-Performance Switch," *IBM Systems J.*, vol. 34, no. 2, pp. 185-204, Feb. 1995.
- [5] J. Beecroft, M. Homewood, and M. McLaren, "Meiko CS-2 Interconnect Elan-Elite Design," *Parallel Computing*, vol. 20, no. 10-11, pp. 1627-1638, Nov. 1994.
- [6] Intel Corporation, *Paragon XP/S Product Overview*. 1991.
- [7] Cray Research, Inc., *Cray T3D System Architecture*. 1993.
- [8] ANSI, Inc., *High-Performance Parallel Interface-6400 Mb/s Physical Layer (HIPPI-6400-PH)*. June 1999.
- [9] N.J. Boden et al., "Myrinet: A Gigabit-per-Second Local Area Network," *IEEE Micro*, pp. 29-35, Feb. 1995.
- [10] Y. Tamir and G.L. Frazier, "Dynamically Allocated Multi-Queue Buffers for VLSI Communication Switches," *IEEE Trans. Computers*, vol. 41, no. 6, pp. 725-737, June 1992.
- [11] J. Ding and L.N. Bhuyan, "Evaluation of Multi-Queue Buffered Multistage Interconnection Networks under Uniform and Non-Uniform Traffic Patterns," *Int'l J. Systems Science*, vol. 28, no. 11, 1997.
- [12] W.J. Dally, "Virtual Channel Flow Control," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 3, pp. 194-205, Mar. 1992.
- [13] H. Sethu, H. Shi, S.S. Kanhere, and A.B. Parekh, "A Round-Robin Scheduling Strategy for Reduced Delays in Wormhole Switches with Virtual Lanes," *Proc. Int'l Conf. Comm. in Computing*, June 2000.
- [14] A. Demers, S. Keshav, and S. Shenker, "Design and Analysis of a Fair Queuing Algorithm," *Proc. ACM SIGCOMM*, pp. 1-12, Sept. 1989.
- [15] S. Keshav, "On the Efficient Implementation of Fair Queuing," *J. Internetworking Research and Experience*, vol. 2, no. 3, pp. 3-26, Sept. 1990.



- [16] A.K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control—The Single Node Case," *Proc. IEEE INFOCOM*, pp. 915-924, May 1992.
- [17] S.J. Golestani, "A Self-Clocked Fair Queuing Scheme for Broadband Applications," *Proc. IEEE INFOCOM*, pp. 636-646, June 1994.
- [18] S. Floyd and V. Jacobson, "Link-Sharing and Resource Management Models for Packet Networks," *IEEE Trans. Networking*, vol. 3, no. 4, pp. 365-386, Aug. 1995.
- [19] S. Floyd, "Notes on Class-Based-Queueing and Guaranteed Service," Unpublished Notes: <http://www.aciri.org/floyd/cbq.html>, July 1995.
- [20] G. Parulkar, H. Adiseshu, and G. Varghese, "A Reliable and Scalable Striping Protocol," *Proc. ACM SIGCOMM*, pp. 131-141, Aug. 1996.
- [21] J.C.R. Bennett and H. Zhang, "WF<sup>2</sup>Q: Worst-Case Fair Weighted Fair Queueing," *Proc. IEEE INFOCOM*, pp. 120-128, Mar. 1996.
- [22] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round-Robin," *IEEE Trans. Networking*, vol. 4, no. 3, pp. 375-385, June 1996.
- [23] P. Goyal, H.M. Vin, and H. Cheng, "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," *IEEE Trans. Networking*, vol. 5, no. 5, pp. 690-704, Oct. 1997.
- [24] S. Keshav, *An Engineering Approach to Computer Networks*. Reading, Mass.: Addison-Wesley, 1997.
- [25] J.A. Cobb, M.G. Gouda, and A. El-Nahas, "Time-Shift Scheduling-Fair Scheduling of Flows in High-Speed Networks," *IEEE Trans. Networking*, vol. 6, no. 3, pp. 274-285, June 1998.
- [26] D. Stiliadis and A. Varma, "Efficient Fair Queueing Algorithms for Packet-Switched Networks," *IEEE Trans. Networking*, vol. 6, no. 2, pp. 175-185, Apr. 1998.
- [27] K. Thompson, G.J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network*, vol. 11, no. 6, pp. 10-23, Nov./Dec. 1997.
- [28] L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications*. New York: Wiley Interscience, 1975.



**Salil S. Kanhere** (S '00) received the BE degree in electrical engineering from the University of Bombay, Bombay, India, in June 1998. He is currently pursuing the PhD degree in the Department of Electrical and Computer Engineering at Drexel University, Philadelphia, Pennsylvania. His research interests include design, analysis, and implementation of scheduling disciplines in packet switched networks. He is a student member of the IEEE.



**Harish Sethu** (M '99) obtained the BTech degree in electronics and communication engineering from Indian Institute of Technology (IIT), Chennai, in 1988. He received the PhD degree in electrical engineering from Lehigh University in 1992. He worked as an Advisory Development Engineer/Scientist at IBM Corporation for six years, during which he contributed to the hardware, software, and system-level design of three generations of the RS/6000 SP family of high-performance parallel computers. He joined the Department of Electrical and Computer Engineering at Drexel University in 1998 as an assistant professor. He has been awarded three US patents. He is also a recipient of the US National Science Foundation CAREER award. His current research interests include quality-of-service in computer networks and the architecture of switches and routers. He is a member of the IEEE.



**Alpa B. Parekh** received the BE degree in electrical engineering from the University of Bombay, Bombay, India, in June 1998 and the MS degree in electrical engineering from Drexel University, Philadelphia, PA in June 2000. She currently works as a software engineer in the Networks and Protocol Development Department at Lockheed Martin Global Telecommunications in Clarksburg, MD.

► For more information on this or any computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.

# Low-latency guaranteed-rate scheduling using Elastic Round Robin

Salil S. Kanhere\*, Harish Sethu

*Department of Electrical and Computer Engineering, Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104, USA*

Received 29 January 2001; revised 5 December 2001; accepted 5 December 2001

---

## Abstract

Packet scheduling algorithms in switches and routers will likely play a critical role in providing the Quality-of-Service (QoS) guarantees required by many real-time multimedia applications. Elastic Round Robin (ERR), a recently proposed fair scheduling discipline designed for best-effort traffic, is very efficient with an  $O(1)$  dequeuing complexity and, in addition, has better fairness characteristics than other algorithms of equivalent complexity. In this paper, we analyze ERR for guaranteed-rate services, and obtain an upper bound on its latency. We further show that the bound obtained in this paper is tight. Our analysis shows that ERR, in comparison to other scheduling disciplines of equivalent complexity, also has significantly better latency properties. The combination of fairness, efficiency and low-latency makes ERR an attractive scheduling discipline for both best-effort and guaranteed-rate services. © 2002 Elsevier Science B.V. All rights reserved.

**Keywords:** Elastic Round Robin; Fair queuing; Guaranteed-rate scheduling; Latency

---

## 1. Introduction

Future high-speed packet-switched networks are expected to support a variety of services beyond the best-effort service available in the Internet today. A number of new applications such as distance learning and multimedia teleconferencing rely on the ability of the network to guarantee such services. For example, such applications would expect the network to ensure that each flow of traffic receives its fair share of the bandwidth and is able to provide performance guarantees such as an upper bound on the end-to-end delay. This requires a Quality-of-Service (QoS) mechanism to efficiently apportion, allocate and manage limited resources among competing users. An important component of such a mechanism is the traffic scheduling algorithm typically used at the output links of switches and routers.

The function of a packet scheduler at an output link is to select the next packet for transmission from among the packets awaiting transmission through the output link. Some of the most important and desirable properties of a scheduling discipline are fairness, efficiency and low-latency, as described below.

- **Fairness.** The available link bandwidth must be distributed among the flows sharing the link in a fair manner.

This ensures that the performance achieved by a flow is not affected when a possibly misbehaving flow tries to transmit packets at a rate faster than its fair share. In this paper, we use the classic notion of fairness given by the max–min fair share policy [1].

- **Latency.** For guaranteed-rate services, the latency should be measured as the length of time it takes a new flow to begin receiving service at the guaranteed rate. This latency is directly related to the amount of playback buffering required at the receiver.
- **Efficiency.** In high-speed networks with large numbers of active flows, the time available for a scheduler to make its scheduling decision is very small. Hence, it is desirable that the time to enqueue a received packet or to dequeue a packet for transmission is as independent as possible of the number of flows sharing the output link. A per-packet work complexity of  $O(1)$  is most desirable.

Scheduling algorithms can be broadly classified into two categories—sorted-priority schedulers and frame-based schedulers. Sorted-priority schedulers maintain a global variable known as the virtual time or the system potential function. A sorted-priority scheduler then uses this variable to compute the timestamp for each packet indicating the relative priority of the packet for transmission over the output link. The packets are then scheduled in increasing order of their timestamps. Examples of sorted-priority schedulers are Weighted Fair Queuing (WFQ) [2,3], Self-Clocked Fair Queuing (SCFQ) [4], Start-Time Fair Queuing

---

\* Corresponding author.

E-mail address: salil@ece.drexel.edu (S.S. Kanhere).

(SFQ) [5], Frame-Based Fair Queuing<sup>1</sup> (FFQ) [6] and Worst-Case Fair Weighted Fair Queuing (WF<sup>2</sup>Q) [7]. The sorted-priority schedulers differ in the manner in which they calculate the global virtual time function. There are two major costs associated with the implementation of sorted-priority schedulers:

1. The complexity of computing the system virtual time: For WFQ, the worst-case complexity is  $O(n)$  where  $n$  is the number of flows sharing the same output link. However, in a number of schedulers such as SCFQ, SFQ and FFQ proposed in recent years, the complexity of computing the virtual time is  $O(1)$ .
2. The complexity of maintaining a sorted list of packets based on their timestamps, and the complexity of computing the maximum or the minimum in this list prior to each packet transmission. For  $n$  flows the work complexity of the scheduler prior to each packet transmission is  $O(\log n)$ .

In frame-based schedulers such as Deficit Round Robin (DRR) [8] and Elastic Round Robin (ERR) [9], the scheduler visits all the non-empty queues in a round robin order. During each service opportunity of a flow, the intent of such a scheduler is to provide to the flow an amount of service proportional to its fair share of the bandwidth. The frame-based schedulers do not maintain a global virtual time function and also do not require any sorting among the packets available for transmission. This reduces the implementation complexity of frame-based scheduling disciplines to  $O(1)$ , making them attractive for implementation in routers, and especially so, in hardware switches.

Elastic Round Robin (ERR) [9] is a recently proposed frame-based scheduling discipline for best-effort traffic, that achieves very good efficiency with a low per-packet work complexity of  $O(1)$  with respect to the number of flows. In addition, it has better fairness properties than other schedulers of equivalent work complexity such as DRR. In this paper, we show that ERR can also be easily adapted for scheduling guaranteed-rate connections, and that it belongs to the class of Latency-Rate ( $\mathcal{LR}$ ) Servers [10], with a latency bound significantly lower than those of other scheduling disciplines of comparable work complexity. These properties of ERR make it an attractive scheduling discipline for both best-effort and guaranteed-rate services.

Section 2 of this paper briefly describes the concept of *Latency-Rate ( $\mathcal{LR}$ ) Servers*, a general class of schedulers proposed by Stiliadis and Verma [10]. In Section 3, we present a weighted version of ERR for serving guaranteed-rate flows. In Section 4, we evaluate the latency

bound of ERR and prove that it belongs to the class of  $\mathcal{LR}$  servers. Section 5 compares the fairness, latency and work complexity of ERR with other guaranteed-rate scheduling disciplines. A tabulated summary of the properties is also provided. Finally, Section 6 concludes the paper.

## 2. $\mathcal{LR}$ servers and related background

In deriving an upper bound on the latency of ERR, we use the concept of  $\mathcal{LR}$  servers first proposed in Ref. [10]. We define a flow as *active* during an interval of time, if at all instants of time during this interval, it has at least one packet awaiting service or being served. We now define the notion of a *busy period*, an essential component of the concept of  $\mathcal{LR}$  servers.

**Definition 1.** A *busy period* of a flow is defined as the maximal time interval during which the flow is active if it is served at exactly its reserved rate.

Let  $Arrived_i(t_1, t_2)$  denote the total number of bits of flow  $i$  that arrive at the scheduler during the time interval  $(t_1, t_2)$ . Consider an interval of time  $(\tau_1, \tau_2)$  which represents a busy period for flow  $i$ . Then for any time interval  $(\tau_1, t)$  such that  $t \in (\tau_1, \tau_2)$ , the number of bits that arrive during this interval is greater than or equal to the number of bits that would exit the scheduler if the flow received service at its reserved rate,  $\rho_i$ . In other words, for all  $t \in (\tau_1, \tau_2)$ ,

$$Arrived_i(\tau_1, t) \geq \rho_i(t - \tau_1)$$

The definition of the busy period supposes that flow  $i$  is served at the constant reserved rate, and therefore, depends only on the reserved rate of the flow and the packet arrival pattern of the flow. An active period of a flow, however, reflects the actual behavior of the scheduler where the instantaneous service offered to flow  $i$  varies according to the number of active flows. If during a busy period of flow  $i$ , the instantaneous service rate offered to flow  $i$  is greater than the allocated rate, then the flow may cease to be active. Thus, a busy period of a flow may include multiple active periods for that flow. Note that the start of a busy period of a flow is always caused by the arrival of a packet belonging to the flow.

The following definitions lead to a formal notion of latency in the case of guaranteed-rate servers. The reader is referred to Ref. [10] for a more detailed discussion.

**Definition 2.** Define  $Sent_i(t_1, t_2)$  as the amount of service received by flow  $i$  during the interval  $(t_1, t_2)$ .

**Definition 3.** Let time instant  $\alpha_i$  represent the start of a certain busy period for flow  $i$ . Let  $t > \alpha_i$  be such that the flow is continuously busy during the time interval  $(\alpha_i, t)$ .

<sup>1</sup> Note that frame-based fair queuing, in spite of its name, is actually a sorted-priority scheduling discipline. The algorithm uses a framing approach similar to that used in frame-based schedulers to update the state of the system. However, as in sorted-priority schedulers, packets are transmitted based on their timestamps.

Define  $S_i(\alpha_i, t)$  as the number of bits belonging to packets in flow  $i$  that arrive after time  $\alpha_i$  and are scheduled during the time interval  $(\alpha_i, t)$ .

Note that  $Sent_i(\alpha_i, t)$  is not necessarily equal to  $S_i(\alpha_i, t)$ . This is because, during this interval of time, the scheduler may still be serving packets that arrived during a previous busy period.  $S_i(\alpha_i, t)$ , therefore, is not necessarily the same as the total number of bits scheduled from flow  $i$  in this interval. We are now prepared to present the definition of latency in  $\mathcal{LR}$  servers.

**Definition 4.** The latency of a flow is defined as the minimum non-negative constant  $\Theta_i$  that satisfies the following for all possible busy periods of the flow,

$$S_i(\alpha_i, t) \geq \max\{0, \rho_i(t - \alpha_i - \Theta_i)\} \quad (1)$$

As defined in Ref. [10], a scheduler which satisfies Eq. (1) for some non-negative constant value of  $\Theta_i$  is said to belong to the class of  $\mathcal{LR}$  servers.

The above definition captures the fact that the latency of a guaranteed-rate scheduler should not merely be the time it takes for the first packet of a flow to get scheduled, but should be a measure of the cumulative time that a flow has to wait until it begins receiving service at its guaranteed rate.

### 3. Weighted elastic round robin

ERR was originally proposed as a fair and efficient scheduling discipline for use in wormhole networks, popular in interconnection networks of parallel systems. The reader is referred to Ref. [9] for a detailed discussion of ERR for best-effort traffic. In this paper, we present a weighted version of ERR for guaranteed-rate services.

Consider an output link of transmission rate  $r$ , access to which is controlled by the ERR scheduler. Let  $n$  be the total number of flows and let  $\rho_i$  be the reserved rate for flow  $i$ . Let  $\rho_{\min}$  be the smallest of the reserved rates. Note that since all the flows share the same output link, a necessary constraint is that the sum of the reserved rates be no more than the transmission rate of the output link. In order that each flow receives service proportional to its guaranteed rate, the ERR scheduler assigns a weight to each flow. The weight assigned to flow  $i$ ,  $w_i$ , is given by,

$$w_i = \frac{\rho_i}{\rho_{\min}} \quad (2)$$

Note that for any flow  $i$ ,  $w_i \geq 1$ .

The ERR scheduler maintains a linked list of the active flows, called the *ActiveList*. When a packet arrives, and if it belongs to a new flow not in the *ActiveList*, the flow is added

to the tail of the *ActiveList*. The ERR scheduler always serves a packet from the flow at the head of the *ActiveList*.

An important and distinct feature of ERR is its definition of a round. We define a round as one round robin iteration consisting *only* of the service visits to all the flows included in the *ActiveList* at the start of the round. For example, a new flow that becomes active while a round is in progress is immediately added to the *ActiveList*, but is served only in the next round. Each flow receives no more than one service opportunity in each round. The scheduler selects the flow at the head of the *ActiveList* for service and calculates its *Allowance*, defined as the number of bits that the flow can transmit in the current service opportunity. In an ERR scheduler, this allowance is *elastic*, i.e. a flow may exceed its allowance during a round. Let  $A_i(s)$  represent the allowance for flow  $i$  during round  $s$ , and let  $Sent_i(s)$  be the number of bits actually scheduled from flow  $i$  during this round. The ERR scheduler will not begin transmission of the next packet in the queue for flow  $i$ , unless  $Sent_i(s)$  is less than  $A_i(s)$ . When  $Sent_i(s)$  becomes greater than  $A_i(s)$  during the transmission of a packet, the scheduler does not pre-empt the transmission but instead, waits for the end of the transmission and then begins the service opportunity for the next flow in the *ActiveList*. At this point, if flow  $i$  is still active, it is added back at the tail end of the list.

The number of excess bits of a flow sent in addition to its allowance during a service opportunity is recorded in the *surplus count*. Let  $SC_i(s)$  represent the surplus count of flow  $i$  in round  $s$ . Following the service of flow  $i$  during the  $s$ th round, its surplus count is calculated as:

$$SC_i(s) = Sent_i(s) - A_i(s) \quad (3)$$

$MaxSC(s)$  is defined as the largest surplus count among all the flows served in round  $s$ . The allowance for each flow is calculated using the  $MaxSC$  value in the previous round, as follows:

$$A_i(s) = w_i(1 + MaxSC(s - 1)) - SC_i(s - 1) \quad (4)$$

Note that the allowance of each flow during the next round depends on the amount by which other flows exceed their allowances in the current round. Each flow seeks to catch up with other flows by correspondingly increasing its allowance for the next round. The allowances, however, are bounded as follows. Let  $m$  be the size in bits of the largest packet that is *actually* served during the execution of a scheduling algorithm. It is readily observed that since no new packet of a flow is scheduled for transmission once the flow has exceeded its allowance, the surplus count is always less than the size of the last packet served. More generally, for any flow  $i$  and round  $s$ ,

$$0 \leq SC_i(s) \leq m - 1 \quad (5)$$

$$0 \leq MaxSC(s) \leq m - 1 \quad (6)$$

Note that, as opposed to DRR [8], ERR does not use a pre-determined *quantum* as the service each flow is

expected to receive during a service opportunity. This gives ERR a couple of important advantages over DRR as well as certain other schedulers such as Surplus Round Robin (SRR) [11–13] which are based on the principle of assigning a quantum to each flow. Firstly, the fairness and the latency bounds of ERR depend only on the sizes of the packets that actually arrive in the flow rather than on the size of the largest packet that may potentially arrive at the scheduler. This significantly improves the latency and fairness bounds since, in most networks including the Internet, the average packet size is much smaller than the maximum possible packet size [14,15]. Secondly, unlike DRR and SRR, ERR does not require knowledge of the upper bound on the transmission time of a packet to achieve a work complexity of  $O(1)$ , rendering it easier to adapt to networks in which the scheduler cannot presume knowledge of the size of a packet prior to its scheduling action. This is important in wormhole networks where the transmission time of a packet depends not just on the length of the packet but also on the congestion in the network. This is also important in other contexts such as in an ATM network transmitting IP packets over AAL5, where the end of the packet is not known until the last ATM cell of the packet arrives.

#### 4. Latency bound of ERR

The analysis of the latency of ERR is facilitated by the following result, stated below as Lemma 1 and proved in Ref. [10]. This result allows one to obtain a bound on the latency achieved by a flow as given by Definition 3 by considering only the active periods of a flow.

**Lemma 1.** *Let  $\tau_i$  be an instant of time when flow  $i$  becomes active. Let  $t > \tau_i$  be some instant of time such that the flow is continuously active during the time interval  $(\tau_i, t)$ . Let  $\Theta'_i$  be the smallest non-negative number such that the following is satisfied for all  $t$ .*

$$\text{Sent}_i(\tau_i, t) \geq \max\{0, \rho_i(t - \tau_i - \Theta'_i)\} \quad (7)$$

*Even though  $(\tau_i, t)$  may not be a continuously busy period for flow  $i$ , the latency as defined by Eq. (1) is bounded by  $\Theta'_i$ .*

We now proceed to derive a bound on the latency of the ERR scheduler, using the lemma above. Note that the instant of time when a flow  $i$  becomes active,  $\tau_i$ , may or may not coincide with the start of the round robin service opportunity of some other flow. In the following, we prove that a tight upper bound on the latency of the ERR scheduler can be obtained by considering  $\tau_i$  belonging to only a subset of all possible time instants. This subset is the set of all time instants that coincide with the beginning or the end of the service opportunity of flows.

**Definition 5.** Define  $\mathbf{T}$  as the set of all time instants, during an execution of the ERR algorithm, at which the scheduler ends serving one flow and begins serving another. Define  $\mathbf{T}_i$  as the set of all time instants at which the scheduler begins serving flow  $i$ . Note that the set  $\mathbf{T}$  is the union of  $\mathbf{T}_i$  for all flows  $i$ , that are served during the execution of the scheduler.

**Lemma 2.** *The latency experienced by flow  $i$  in an ERR scheduler will reach its upper bound,  $\Theta'_i$ , only if the time instant,  $\tau_i$ , at which flow  $i$  becomes active, belongs to the set  $\mathbf{T}$ .*

**Proof.** Assume that flow  $i$  becomes active at time instant  $\tau_i$ . Let  $(t_1, t_2)$ ,  $t_1 \leq \tau_i < t_2$  be the time interval during which some flow  $j \neq i$  receives its round robin service opportunity. Consider the case when  $\tau_i$  does not coincide with the start of the service opportunity of flow  $i$ , i.e.  $\tau_i > t_1$ . Now, the time interval  $(t_1, \tau_i)$ , which is a part of the round robin service opportunity of flow  $j$ , will not contribute to the latency experienced by flow  $i$ . On the other hand, consider the case when  $\tau_i$  coincides with  $t_1$ , the start of the service opportunity of flow  $j$ . Now, the time for which flow  $i$  has to wait before receiving any service will include the entire time interval  $(t_1, t_2)$  during which flow  $j$  receives its round robin service opportunity. Clearly, the latency experienced by flow  $i$  is always greater when  $\tau_i$  coincides with the start of the service opportunity of some other flow. The statement of the lemma follows from this observation.

□

From Lemma 2, in deriving an upper bound on the latency experienced by flow  $i$ , therefore, one needs to only consider  $\tau_i$  such that  $\tau_i \in \mathbf{T}$ . The following lemma further limits the cases we need to consider in deriving the upper bound. Note that, in proving the upper bound, we need to only consider the intervals of time over which Eq. (7) is an equality. In fact, the upper bound on the latency is achieved at time instants  $t$  when  $\text{Sent}_i(\tau_i, t) = \rho_i(t - \tau_i - \Theta'_i)$ . The following lemma provides a simple condition on  $t$  in order to achieve the equality in Eq. (7).

**Lemma 3.** *If flow  $i$  becomes active at time instant  $\tau_i$ , then there exists some  $t \in \mathbf{T}_i$  such that the flow remains active during the interval  $(\tau_i, t)$ , and*

$$\text{Sent}_i(\tau_i, t) = \rho_i(t - \tau_i - \Theta'_i)$$

**Proof.** Note that, if  $\text{Sent}_i(\tau_i, t) = \rho_i(t - \tau_i - \Theta'_i)$ , then the flow experiences the worst-case latency at time instant  $t$ . Consider any two consecutive time instants  $t_1$  and  $t_2$  which both belong to  $\mathbf{T}_i$ . Consider an instant of time  $t$ ,  $t_1 < t < t_2$ .

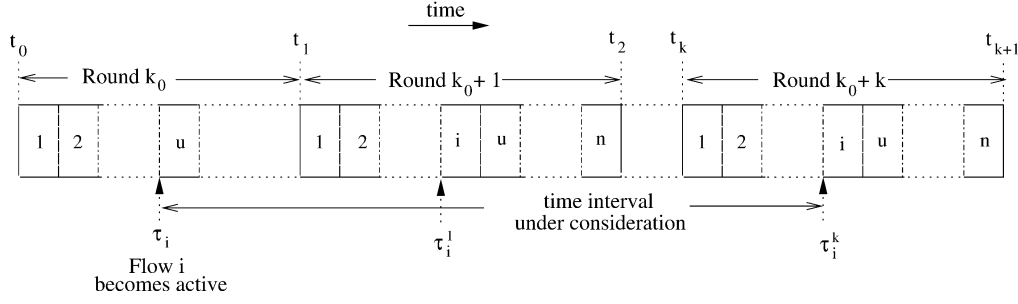


Fig. 1. An illustration of the time interval under consideration.

*Case 1:* Let  $t$  be such that flow  $i$  is receiving service at time instant  $t$ .

During the interval  $(t_1, t)$ , the amount of service received by the flow is  $r(t - t_1)$ , where  $r$  is the rate of the link. Clearly, during this time, the flow is receiving service at the guaranteed rate or higher. Therefore, the worst-case latency experienced by the flow until any time in the interval  $(t_1, t)$  is no worse than that experienced by it until time  $t_1 \in \mathbf{T}_i$ .

*Case 2:* Let  $t$  be such that some flow other than  $i$  is receiving service at time instant  $t$ .

During the interval  $(t, t_2)$ , flow  $i$  receives no service at all. Therefore, the latency experienced by the flow increases after time  $t$  but only until time  $t_2$ . Thus, the worst-case latency experienced by the flow until any time in the interval  $(t, t_2)$  is no worse than that experienced by it until time  $t_2 \in \mathbf{T}_i$ .

The above two cases illustrate that the worst-case latency experienced by a flow during an interval  $(t_1, t_2)$  is equal to the latency experienced by the flow until either time  $t_1$  or time  $t_2$ . Extrapolating to all intervals between consecutive time instants that belong to  $\mathbf{T}_i$ , the statement of the lemma is proved.

□

**Theorem 1.** *The ERR scheduler belongs to the class of  $\mathcal{LR}$  servers, with an upper bound on the latency  $\Theta_i$  for flow  $i$  given by,*

$$\Theta_i \leq \frac{(W - w_i)m + (n - 1)(m - 1)}{r} \quad (8)$$

where  $n$  is the total number of active flows and  $W$  is the sum of the weights of all the flows.

**Proof.** From Lemma 1, we know that the latency of the ERR scheduler as defined by Eq. (1) is bounded by  $\Theta'_i$ . Hence we will prove the theorem by showing that  $\Theta'_i = ((W - w_i)m + (n - 1)(m - 1))/r$ . By the statement of Lemma 2, this upper bound on the latency is reached only if the time instant at which flow  $i$  becomes active,  $\tau_i$ , belongs to  $\mathbf{T}$ . Therefore, in seeking the upper bound on

the latency, we may assume that flow  $i$  becomes active at exactly the instant that some other flow begins receiving service.

Let  $\tau_i^k$  be the time instant marking the start of the  $k$ th service opportunity of flow  $i$  after it becomes active at time instant  $\tau_i$ . Note that  $\tau_i^k$  belongs to  $\mathbf{T}_i$ . Therefore, from Lemma 3, in order to determine the latency bound as defined by Eq. (1), one needs to only consider intervals of time  $(\tau_i, \tau_i^k)$  for all  $k$ . Fig. 1 shows the time interval under consideration for a given  $k$ .

To prove the statement of the theorem, we first obtain the lower bound on the total service received by flow  $i$  during the time interval under consideration. Then we express the lower bound in the form of Eq. (7) to derive the latency bound.

Note that the time instant  $\tau_i$  may or may not coincide with the end of a round and the start of the subsequent round. Let  $k_0$  be the round which is in progress at time instant  $\tau_i$  or which ends exactly at time instant  $\tau_i$ . Let the time instant  $t_h$  mark the end of round  $(k_0 + h - 1)$  and the start of the subsequent round. For any flow  $j$  during a round  $s$  in the interval under consideration, using Eqs. (3) and (4), we have,

$$Sent_j(s) = w_j(1 + \text{MaxSC}(s - 1)) + SC_j(s) - SC_j(s - 1) \quad (9)$$

As shown in Fig. 1, assume that the time instant when flow  $i$  becomes active coincides with the time instant when some flow  $j$  is about to start its service opportunity during the  $k_0$ th round. Let  $G_a$  denote the set of flows which receive service during the time interval  $(\tau_i, t_1)$ , i.e. after flow  $i$  becomes active and during round  $k_0$ . Similarly, let  $G_b$  denote the set of flows which are served by the ERR scheduler during the time interval  $(t_0, \tau_i)$ , i.e. before flow  $i$  becomes active and during round  $k_0$ . Note that flow  $i$  is not included in either of these two sets since flow  $i$  will receive its first service opportunity only in the  $(k_0 + 1)$ th round. If the time instant  $\tau_i$  coincides with the end of a round, then the set  $G_a$  will be empty and all the  $n - 1$  flows will belong to the set  $G_b$ .

Consider the time interval  $(\tau_i, \tau_i^k)$  during which flow  $i$  receives  $k$  round robin service opportunities. This time interval can be split into three sub-intervals:

(1)  $(\tau_i, t_1)$ : This sub-interval includes the part of the  $k_0$ th

round during which all the flows belonging to the set  $G_a$  will be served by the ERR scheduler. Summing Eq. (9) over all these flows,

$$t_1 - \tau_i \leq \frac{1}{r} \sum_{j \in G_a} \{w_j(1 + \text{MaxSC}(k_0 - 1)) + \text{SC}_j(k_0) - \text{SC}_j(k_0 - 1)\} \quad (10)$$

(2)  $(t_1, t_k)$ : This sub-interval includes  $k - 1$  rounds of execution of the ERR scheduler starting at round  $(k_0 + 1)$ . Consider the time interval  $(t_h, t_{h+1})$  when round  $(k_0 + h)$  is in progress. Summing Eq. (9) over all  $n$  flows,

$$t_{h+1} - t_h \leq \frac{W}{r}(1 + \text{MaxSC}(k_0 + h - 1)) + \frac{1}{r} \sum_{j=1}^n \text{SC}_j(k_0 + h) - \frac{1}{r} \sum_{j=1}^n \text{SC}_j(k_0 + h - 1)$$

Summing the above over  $k - 1$  rounds beginning with round  $k_0 + 1$ ,

$$t_k - t_1 \leq \frac{W}{r}(k - 1) + \frac{1}{r} \sum_{j=1}^n (\text{SC}_j(k_0 + k - 1) - \text{SC}_j(k_0)) + \frac{W}{r} \sum_{h=1}^{k-1} \text{MaxSC}(k_0 + h - 1) \quad (11)$$

(3)  $(t_k, \tau_i^k)$ : This sub-interval includes the part of the  $(k_0 + k)$ th round during which all the flows belonging to the set  $G_b$  will be served by the ERR scheduler. Summing Eq. (9) over all these flows,

$$\tau_i^k - t_k \leq \frac{1}{r} \sum_{j \in G_b} w_j(1 + \text{MaxSC}(k_0 + k - 1)) + \frac{1}{r} \sum_{j \in G_b} \text{SC}_j(k_0 + k) - \frac{1}{r} \sum_{j \in G_b} \text{SC}_j(k_0 + k - 1) \quad (12)$$

Combining Eqs. (10)–(12) and using Eq. (5) we have,

$$\begin{aligned} \tau_i^k - \tau_i &\leq \frac{W}{r}(k - 1) + \frac{W - w_i}{r} \\ &+ \frac{1}{r} \sum_{j \in G_a} w_j \text{MaxSC}(k_0 - 1) \\ &+ \frac{W}{r} \sum_{h=1}^{k-1} \text{MaxSC}(k_0 + h - 1) \\ &+ \frac{1}{r} \sum_{j \in G_b} w_j \text{MaxSC}(k_0 + k - 1) \\ &+ \frac{(n - 1)(m - 1)}{r} + \frac{1}{r} \text{SC}_i(k_0 + k - 1) \end{aligned}$$

Solving for  $(k - 1)$ ,

$$\begin{aligned} (k - 1) &\geq (\tau_i^k - \tau_i) \frac{r}{W} - \frac{W - w_i}{W} \\ &- \frac{1}{W} \sum_{j \in G_a} w_j \text{MaxSC}(k_0 - 1) \\ &- \sum_{h=1}^{k-1} \text{MaxSC}(k_0 + h - 1) \\ &- \frac{1}{W} \sum_{j \in G_b} w_j (\text{MaxSC}(k_0 + k - 1) \\ &- \frac{(n - 1)}{W}(m - 1) - \frac{1}{W} \text{SC}_i(k_0 + k - 1) \end{aligned} \quad (13)$$

Using Eq. (9) over these  $(k - 1)$  rounds of service for flow  $i$ , and since the surplus count of a newly active flow is 0, we get,

$$\begin{aligned} \text{Sent}_i(\tau_i, \tau_i^k) &= \text{SC}_i(k_0 + k - 1) + w_i(k - 1) \\ &+ w_i \sum_{h=1}^k \text{MaxSC}(k_0 + h - 1) \end{aligned} \quad (14)$$

Now, since the reserved rates are proportional to the weights assigned to the flows as given by Eq. (2), and since the sum of the reserved rates is no more than the link rate  $r$ , we have,

$$\rho_i \leq \frac{w_i}{W} r \quad (15)$$

Using the above and using Eq. (13) to substitute for  $(k - 1)$  in Eq. (14), we get,

$$\begin{aligned} \text{Sent}_i(\tau_i, \tau_i^k) &\geq \rho_i(\tau_i^k - \tau_i) - \frac{w_i}{W}(W - w_i) \\ &- \frac{w_i}{W} \sum_{j \in G_a} w_j \text{MaxSC}(k_0 - 1) \\ &- \frac{w_i}{W} \sum_{j \in G_b} w_j \text{MaxSC}(k_0 + k - 1) \\ &- \frac{w_i}{W}(n - 1)(m - 1) - \text{SC}_i(k_0 + k - 1) \left( \frac{w_i}{W} - 1 \right) \end{aligned}$$

Using Eq. (6) and since  $W$  is the sum of the weights of all the  $n$  flows, we have,

$$\begin{aligned} \text{Sent}_i(\tau_i, \tau_i^k) &\geq \rho_i(\tau_i^k - \tau_i) - \frac{w_i}{W}(W - w_i) \\ &- \frac{w_i}{W}(W - w_i)(m - 1) - \frac{w_i}{W}(n - 1)(m - 1) \\ &- \text{SC}_i(k_0 + k - 1) \left( \frac{w_i}{W} - 1 \right) \end{aligned}$$

Table 1

A comparison between scheduling disciplines

Scheduler	Complexity	Fairness	Latency bound for flow $i$
GPS [3]		0	0
Weighted Fair Queuing [2]	$O(n)$	$O(n)$	$\frac{m}{r} + \frac{m}{\rho_i}$
Self-Clocked Fair Queuing [4]	$O(n)$	$2m$	$\frac{(n-1)m}{r} + \frac{m}{\rho_i}$
Virtual Clock [16]	$O(n)$	$\infty$	$\frac{m}{r} + \frac{m}{\rho_i}$
Frame-based Fair Queuing [6]	$O(n)$	$2M + m$	$\frac{m}{r} + \frac{m}{\rho_i}$
Deficit Round Robin [8]	$O(1)$	$M + 2m$	$\frac{(W - w_i)M + (n-1)(m-1)}{r} + (\frac{1}{\rho_i} - \frac{1}{r})(m-1)$
Elastic Round Robin [9]	$O(1)$	$3m$	$\frac{(W - w_i)m + (n-1)(m-1)}{r}$

Using Eq. (15), we have,

$$\begin{aligned}
 Sent_i(\tau_i, \tau_i^k) &\geq \rho_i(t_i^k - \tau_i) \\
 &- \frac{\rho_i}{r}(W - w_i)(m) - \frac{\rho_i}{r}(n-1)(m-1) \\
 &- SC_i(k_0 + k - 1)\left(\frac{w_i}{W} - 1\right)
 \end{aligned}$$

Simplifying further, and noting that the latency bound reaches the upper bound when  $SC_i(k_0 + k - 1)$  equals 0, we get,

$$\begin{aligned}
 Sent_i(\tau_i, \tau_i^k) &\geq \max\left\{0, \rho_i\left(\tau_i^k - \tau_i - \frac{(W - w_i)m + (n-1)(m-1)}{r}\right)\right\} \\
 &\quad (16)
 \end{aligned}$$

As discussed earlier, based on Lemmas 2 and 3, flow  $i$  will experience its worst latency during an interval  $(\tau_i, \tau_i^k)$  for some  $k$ . Therefore, from Eq. (16) and Lemma 1, the statement of the theorem is proved.

□

We now proceed to show that the latency bound given by Theorem 1 is tight by illustrating a case where the bound is actually met. Assume that a flow  $i$  becomes active at time instant  $\tau_i$ , which also coincides with the end of a certain round  $k_0$  and the start of the round  $(k_0 + 1)$ . Since other flows in the *ActiveList* will be served first, flow  $i$  becomes backlogged instantly. Assume that for any time instant  $t$ ,  $t \geq \tau_i$ , a total of  $n$  flows, including flow  $i$ , are active. Also, assume that the summation of the reserved rates of all the  $n$  flows equals the output link transmission rate,  $r$ . Hence,  $\rho_i = (w_i/W)r$ . Since flow  $i$  became active at time  $\tau_i$ , its surplus count at the start of round  $(k_0 + 1)$  is 0. Let the surplus count of all the other flows at the start of round  $(k_0 + 1)$  be equal to 0. Assume that, a flow  $p$  which is not active after time  $\alpha_i$  and hence is not included in the  $n$  flows, was active during the  $k_0$ th round. Also assume that flow  $p$

exceeded its allowance by  $(m - 1)$  during its service opportunity in round  $k_0$ , leading to a value of  $MaxSC(k_0)$  equal to  $(m - 1)$ . From Eqs. (5), (6) and (9), any given flow  $j$  can transmit a maximum of  $w_j m + (m - 1)$  bits during a round robin service opportunity. In the worst case, before flow  $i$  is served by the ERR scheduler, each of the other  $(n - 1)$  flows will receive this maximum service. Hence, the cumulative delay until flow  $i$  receives service is given by,

$$\begin{aligned}
 D &= \frac{\left(\sum_{j \neq i} w_j\right)m + (n-1)(m-1)}{r} \\
 &= \frac{(W - w_i)m + (n-1)(m-1)}{r}
 \end{aligned}$$

Noting that  $S_i(\alpha_i, \alpha_i + D)$  equals zero, it is readily verified that the bound is exactly met at time  $t = \alpha_i + D$ .

## 5. Comparison to other schedulers

Table 1 summarizes the work complexity, fairness and latency bounds of several guaranteed-rate scheduling disciplines that belong to the class of  $\mathcal{LR}$  servers. We measure fairness using a well-known and widely used metric, known as the Relative Fairness Bound (RFB) [4]. The RFB is defined as the maximum difference in the normalized service received by any two active flows over all possible intervals of time. In this table,  $M$  is the size of the largest packet that may potentially arrive during the execution of a scheduling algorithm. Recall that  $m$  is the size of the largest packet that *actually* arrives during the execution of the scheduler. Typically,  $M \gg m$ , since in most networks including the Internet, the vast majority of the packets are of much smaller size than the maximum possible size of a packet [14,15]. The properties of all the scheduling disciplines in Table 1 except ERR and DRR are derived in Ref. [17]. Note that the latency bound of DRR stated here is tighter than the



one derived in Ref. [17]. The derivation of this bound is similar to the proof of Theorem 1.

The GPS scheduler visits each active flow in a round-robin fashion, and serves an infinitesimally small amount of data proportional to the reserved rate of the flow [3]. Using this fluid model, the GPS scheduler is able to ensure that over any interval of time however small, the normalized difference between the service received by any two active flows is exactly zero. The RFB of GPS, therefore, is zero. The latency of GPS is also zero since a newly active flow begins receiving service instantaneously at the guaranteed rate. Recall that GPS is an ideal but not an implementable scheduler.

WFQ [2], FFQ [6] and Virtual Clock [16] have a low value of the latency bound. However, the work complexity of these schedulers is greater than those of ERR and DRR.

Virtual Clock has an RFB of infinity and therefore, cannot be considered to be a fair scheduler. The RFB of the WFQ scheduler is  $O(n)$  and thus, has better fairness (the exact expression for the RFB may be found in Ref. [17]). FFQ, on the other hand, has a still lower value of the RFB and thus is significantly more fair. However, FFQ requires periodic re-calibration of the virtual time, and also has a work complexity of  $O(n)$  rendering it less efficient than ERR or DRR. In fact, only ERR and DRR are efficient with an  $O(1)$  work complexity. Table 1 shows that ERR has better fairness properties and a lower latency bound than DRR does, especially considering that  $M$  is typically much greater than  $m$ .

## 6. Conclusion

Elastic Round Robin (ERR), a recently proposed fair, efficient and easily implementable scheduling discipline was designed to satisfy the unique needs of wormhole switching, popular in interconnection networks of parallel systems. It may also be readily adapted for fair scheduling of best-effort traffic in the Internet. In this paper, we present a weighted version of ERR for serving guaranteed-rate flows. We prove that ERR belongs to the class of  $\mathcal{LR}$  servers and evaluate the upper bound on the latency experienced by a flow served by an ERR scheduler. We also show that the bound obtained in this paper is tight.

Our analysis reveals that ERR has better fairness characteristics and a significantly better latency bound in comparison to other scheduling disciplines of equivalent complexity such as DRR. While fairness is an intuitively desirable goal, its practical relevance is in the bound on the latency that fair schedulers are able to provide. This latency, as defined for  $\mathcal{LR}$  servers in Ref. [10], has a direct bearing on the size of the playback buffers needed at the receivers for real-time communications. This paper

shows that, with its low-latency bound, ERR is an attractive scheduling discipline for both best-effort and guaranteed-rate traffic.

## Acknowledgements

This work was supported in part by NSF CAREER grant CCR-9984161 and US Air Force Contract F30602-00-2-0501.

## References

- [1] S. Keshav, An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network, Addison-Wesley, Reading, MA, 1997 Ch 9: Scheduling, pp. 209–261.
- [2] A. Demers, S. Keshav, S. Shenker, Design and analysis of a fair queuing algorithm, Proceedings of ACM SIGCOMM, Austin, 1989, pp. 1–12.
- [3] A.K. Parekh, R.G. Gallager, A generalized processor sharing approach to flow control the single node case, Proceedings of IEEE INFOCOM, Florence, Italy, 1992, pp. 915–924.
- [4] S.J. Golestani, A self-clocked fair queuing scheme for broadband applications, Proceedings of IEEE INFOCOM, Toronto, Canada, 1994, pp. 636–646.
- [5] V.P. Goyal, H.M. Vin, H. Cheng, Start-time fair queuing: a scheduling algorithm for integrated services packet switched networks, IEEE Transactions of Networking 5 (5) (1997) 690–704.
- [6] D. Stiliadis, A. Verma, Efficient fair queuing algorithms for packet-switched networks, IEEE Transactions on Networking 6 (2) (1998) 175–185.
- [7] J.C.R. Bennett, H. Zhang, WF2Q: Worst-case fair weighted fair queuing, Proceedings of IEEE INFOCOM, San Francisco, CA, 1996, pp. 120–128.
- [8] M. Shreedhar, G. Varghese, Efficient fair queuing using deficit round-robin, IEEE Transactions on Networking 4 (3) (1996) 375–385.
- [9] S.S. Kanhere, A. Parekh, H. Sethu, Fair and efficient packet scheduling in wormhole net works, Proceedings of the International Parallel and Distributed Processing Symposium, Cancun, Mexico, 2000, pp. 623–631.
- [10] D. Stiliadis, A. Verma, Latency-rate servers: a general model for analysis of traffic scheduling algorithms, IEEE Transactions on Networking 6 (5) (1998) 611–624.
- [11] S. Floyd, Notes on class-based-queuing and guaranteed service, <http://www.aciri.org/floyd/cbq.html>.
- [12] S. Floyd, V. Jacobson, Link-sharing and resource management models for packet networks, IEEE Transactions on Networking 3 (4) (1995) 365–386.
- [13] G.P.H. Adiseshu, G. Varghese, A reliable and scalable striping protocol, Proceedings of ACM SIGCOMM, Palo Alto, CA, 1996, pp. 131–141.
- [14] K. Thompson, G. Miller, R. Wilder, Widearea internet traffic patterns and characteristic, IEEE Network (1997) 10–23.
- [15] I. Widjaja, A.I. Elwalid, Performance issues in vc-merge capable switches for multiprotocol label switching, IEEE Journal on Selected Areas in Communications 17 (6) (1999) 1178–1178.
- [16] L. Zhang, Virtual clock: a new traffic control algorithm for packet switching networks, Proceedings of ACM SIGCOMM, Philadelphia, PA, 1990, pp. 19–29.
- [17] D. Stiliadis, Traffic scheduling in packet-switched networks: analysis, design, and implementation, PhD thesis, University of California, Santa Cruz, June 1996.

# On Scheduling Real-Time Traffic under Controlled Load Service in an Integrated Services Internet

Hongyuan Shi and Harish Sethu

hyshi@io.ece.drexel.edu, sethu@ece.drexel.edu

Department of Electrical and Computer Engineering

Drexel University

3141 Chestnut Street, Philadelphia, PA 19104-2875

**Abstract**— The controlled load service defined within the IETF's Integrated Services architecture for QoS in the Internet requires source points to regulate the traffic while the network provides a soft guarantee on performance. Packets sent in violation of the traffic are marked so that the network may give them lower priority. In this paper, we have defined the requirements of a scheduler serving packets belonging to the controlled load service. Besides efficiency and throughput goals, we define another important requirement to bound the additional delay of unmarked packets caused due to the transmission of marked packets. For any given desired bound  $\alpha$  on this additional delay, we present the  $CL(\alpha)$  scheduler which achieves the bound while also achieving a per-packet work complexity of  $O(1)$ . We also provide analytical proofs of these results on the  $CL(\alpha)$  scheduler. The principle used in this algorithm can also be used to schedule flows with multilevel priorities, such as in some real-time video streams as well as in other emerging service models of the Internet that mark packets to identify drop precedences.

**Index Terms**— Controlled load service, scheduling, real-time traffic, Integrated Services architectures.

## I. INTRODUCTION

With the rapid evolution of the Internet as a commercial infrastructure, there have been increasing demands for differentiated services based on user and/or application requirements. Real-time traffic, in particular, has stringent requirements of delay and bandwidth guarantees which are not satisfied by the best-effort service provided by much of today's Internet. The guaranteed service model in the Integrated Service architecture defined by the IETF seeks to achieve this by making per-flow reservations using the Resource Reservation Protocol, and then expecting schedulers in the routers to abide by the reservations [2]. One of the challenges in providing these guaranteed services is in the management of reservations and scheduling states corresponding to thousands of traffic flows that may all be active at the same time. Therefore, the Integrated Services framework also specifies a more scalable option called the *controlled load* service [3]. This paper is concerned with design-

ing a scheduler for routers which serve packets belonging to this service category.

Controlled load service is distinguished by the fact that it seeks to provide users with a quality of service similar to that in a lightly loaded or unloaded network, and without requiring or specifying a target upper bound on the delay or loss probabilities. This quality of service is assured through capacity planning and admission control, rather than through per-flow management during packet scheduling and forwarding. The idea behind this service model is that many real-time applications do receive adequate performance and quality of service in a lightly loaded network, eliminating the need for very strict performance guarantees. Each user/application provides an estimate of its traffic specifications,  $T_{spec}$ , and the service provider admits the traffic based on whether or not supporting it would still keep the network "lightly loaded". When the user exceeds the traffic specifications, the service obtained by the excess packets degenerates to the best-effort service. Controlled load service allows a scalable means to achieve the required quality of service since it does not require the network to distinguish between flows beyond the admission control phase.

Packets sent by an application in excess of the  $T_{spec}$  agreed upon by the user and the service provider are marked by a traffic policer at the entry point into the network. As per the definition of the controlled load service, these excess marked packets receive best-effort service while the unmarked packets receive service similar to that in a lightly loaded network. To preserve the generality of our solutions, we refer to these excess packets as simply *marked* packets and the rest as *unmarked* packets. Such marking of packets to indicate their level of importance for dropping policies within the network is also used in the Differentiated Services model defined by the IETF [1], as well as in some applications requiring compressed video transmissions where some packets are more important to the quality of the playback than some other packets.

While scheduling algorithms have been widely studied for best effort traffic as well as for guaranteed services, scheduling strategies for merged packet streams with different service requirements have not been studied within a theoretical

framework. In the following, we discuss the requirements of a scheduler in an Internet router providing controlled load service to real-time traffic with marked and unmarked packets. Section II presents the requirements of such a scheduler and Section III presents the  $CL(\alpha)$  scheduler which satisfies these requirements. Section IV provides brief theoretical analysis. Section V concludes the paper.

## II. REQUIREMENTS

Our primary goal in the design of a scheduler for controlled load service is to preserve the spirit of the controlled load service. Certainly, it would be inappropriate to add implementation complexity to the service by adding per-flow management in the routers. Therefore, it is desirable that the scheduler use some simple discipline such as first-come-first-served, while aggregating packets from all flows into the same queue awaiting service by the scheduler. Note that, in a lightly loaded network with regulated traffic, a first-come-first-served scheduling discipline is expected to be more than adequate. Also, by such a strategy which places all the packets in the same queue, the packets within the same flow, marked or unmarked, are delivered in order.

Secondly, the controlled load service packets do not have a delay or bandwidth specification. Therefore, a scheduler cannot make decisions based on delay requirements as in traditional guaranteed-service schedulers such as virtual clock or weighted fair queueing. Instead, the capacity planning phase, based on the *Tspec* provided by the applications, is responsible for ensuring that the packets can receive a delay approximating that in a lightly loaded network. Therefore, it is safe for the scheduler to assume that the unmarked packets of one flow will not affect the unmarked packets of another flow to the point that the network appears congested to any flow. However, the marked packet arrival characteristics are not part of the *Tspec* and therefore, unregulated. The scheduler does have to ensure that the impact of too many marked packets on the quality-of-service received by the traffic flows is kept under control within a certain acceptable bound. Since delay is the primary QoS parameter for real-time traffic, we can define the scheduler requirement as follows: the scheduler should guarantee that, *for any unmarked packet, the additional delay caused by marked packet transmissions is no more than  $\alpha$* . In other words, if an unmarked packet, in the absence of marked packets, could be forwarded with a delay of  $\Delta$ , then the delay of the same packet in the presence of marked packets should be no more than  $\Delta + \alpha$ . The quantity  $\alpha$  may be defined by the router or may be a negotiated quantity between service providers.

Finally, we do wish to send as many marked packets (best-effort packets) as possible without violating the above requirement on its impact on the delay of unmarked packets. This requirement, while ensuring that unmarked packets are never dropped or delayed beyond a certain point, is non-trivial to

achieve, especially in the absence of per-flow management. Note that in the absence of per-flow tracking and management of packet arrivals, the scheduler cannot predict with sufficient precision the new packet arrival characteristics, and therefore, cannot know whether sending a marked packet at a certain instant of time can be a cause for additional delay for unmarked packets at some later time.

In the following section, we describe the  $CL(\alpha)$  scheduler for Controlled Load service, which meets all of the above requirements, in addition to being simple enough to implement with an  $O(1)$  per-packet dequeuing complexity.

## III. THE $CL(\alpha)$ SCHEDULER

The  $CL(\alpha)$  scheduler maintains a single first-come-first-served queue for all arriving packets. Marked as well as unmarked packets are all added to the tail of the same queue in order of their arrival times. The  $CL(\alpha)$  scheduler removes packets from the head of the queue for service, but marked packets may be dropped at the server if sending it introduces an unacceptable extra delay to unmarked packets. The scheduler's responsibility is to ensure that the delay of an unmarked packet does not increase by more than  $\alpha$  due to the transmission of a marked packet.

We define extra delay of an unmarked packet as the additional delay caused by the transmission of marked packets. Obviously, when a marked packet is scheduled for transmission, all the unmarked packets in the queue suffer an extra delay. Furthermore, some of the extra delay is also "passed on" to the unmarked packets which arrive after the transmission of the marked packet. This is because, in a first-come-first-served queue, a packet's delay depends on the time when its predecessor is served. Thus, the extra delay caused to one unmarked packet can cause an extra delay to unmarked packets that arrive later as well. However, the extra delay of a newly arrived packet is not merely equal to the extra delay suffered by its predecessor, and can actually be less than that of its predecessor in the queue. This is best illustrated by considering an unmarked packet that arrives during a period of low congestion when the outgoing rate is larger than the sum of the incoming flow rates. However, the unmarked packet ahead of it in the queue, i.e., the predecessor packet, may have arrived in the queue during a period of heavy congestion when the queue length was large and thus may have a large extra delay associated with it. Assume that the queue length is now reduced to the level that the total transmission time needed for the whole queue is less than the extra delay of this predecessor packet. Under these circumstances, the newly arrived packet will not inherit all of the delay suffered by the predecessor packet, but only part of it.

Since the  $CL(\alpha)$  scheduler needs to control the extra delay of unmarked packets, the system has to keep track of the changes in the extra delay of each unmarked packet. A naive method for this is to simply use an extra delay counter for each unmarked

*Initialize:* (Invoked when the scheduler is initialized)

```
HeadED = 0;
TailedED = 0;
```

*Enqueue:* (Invoked when a packet  $P$  arrives)

```
if (P is unmarked AND EDDQueueIsNotEmpty) then
  if (UMPD < TailedED) then
    AddToEDDQueue(TailedED - UMPD);
    TailedED = UMPD;
  else
    AddToEDDQueue(0);
  end if;
else if (P is unmarked AND EDDQueueIsEmpty) then
  if (the server is transmitting a packet) then
    V = PacketBeingTransmitted;
    if (V is marked)
      HeadED = UMPD;
      TailedED = UMPD;
    else
      if (PreviousHeadED > UMPD) then
        HeadED = UMPD;
        TailedED = UMPD;
      else
        HeadED = PreviousHeadED;
        TailedED = PreviousHeadED;
      end if;
    end if;
  end if;
  AddToEDDQueue(0);
end if;
AddPacketToQueue(P);
```

Fig. 1. Initialization and Enqueueing routines of the  $CL(\alpha)$  scheduler

packet. The scheduler in such a case would have to check each of the extra delay counters before sending a marked packet. Upon sending a marked packet, it would have to update each counter by the transmission time of the marked packet. Obviously, this scheme has a processing delay proportional to the number of unmarked packets in the queue, with the potential to severely limit scheduling efficiency when the queue lengths are large. The  $CL(\alpha)$  scheduler, however, achieves significantly better scalability in terms of the processing delay, with an  $O(1)$  per-packet work complexity. Fig. 1 and 2 present a pseudo-code description of the  $CL(\alpha)$  scheduler.

We denote the first unmarked packet in the queue (i.e., closest to the head of the queue) as the *unmarked head*, and the last unmarked packet in the queue (i.e., closest to the tail) as the *unmarked tail*. In the  $CL(\alpha)$  scheduler, the *extra delay (ED)* of an unmarked packet at a certain instant of time is defined as the cumulative additional delay experienced by the packet because

*Dequeue:*

```
while (QueueIsNotEmpty) do
  P = PacketAtHeadOfQueue;
  if (P is unmarked) then
    Remove EDDHead from EDDQueue;
    if (EDDQueueIsNotEmpty AND
        EDDHead < HeadED) then
      HeadED = HeadED - EDDHead;
    else
      if (EDDQueueIsEmpty) then
        PreviousHeadED = HeadED;
      end if;
      HeadED = 0;
      TailedED = 0;
    end if;
    TransmitPacket(P);
  else
    if (EDDQueueIsNotEmpty) then
      if (HeadED + TxTime(P) ≤ α) then
        HeadED = HeadED + TxTime(P);
        TailedED = TailedED + TxTime(P);
        TransmitPacket(P);
      else
        V = UnmarkedHeadOfQueue;
        Drop all packets ahead of V;
        Remove EDDHead from EDDQueue;
        if (EDDQueueIsNotEmpty AND
            EDDHead < HeadED) then
          HeadED = HeadED - EDDHead;
        else
          if (EDDQueueIsEmpty) then
            PreviousHeadED = HeadED;
          end if;
          HeadED = 0;
          TailedED = 0;
        end if;
        TransmitPacket(V);
      end if;
    else
      TransmitPacket(P);
    end if;
  end while;
```

Fig. 2. Dequeueing routine of the  $CL(\alpha)$  scheduler

of marked packet transmissions, if the packet is scheduled for transmission at exactly that time instant. Instead of keeping a counter for each unmarked packet, the  $CL(\alpha)$  scheduler maintains a record of the extra delay for the unmarked head and tail packets, denoted by *HeadED* and *TailedED*. Both of them are updated whenever a marked packet is transmitted while there

are unmarked packets in the queue. Meanwhile, the scheduler stores a quantity called the *ExtraDelayDifference*, for each unmarked packet in the queue. The extra delay of any packet at any instant of time can be calculated from *HeadED* at that instant of time and the value of the *ExtraDelayDifference* of the packet. In the following, we define the quantity *ExtraDelayDifference* and the procedure to compute the extra delay of any packet.

Suppose all the arriving unmarked packets are labeled as 0, 1, 2, ..., in order of their arrival times. Let  $a_i$  be the arrival time of packet  $i$ . Denote the departure time for packet  $i$  by  $d_i$ , and the extra delay of packet  $i$  at time  $t$  as  $ED_i(t)$ . If  $a_i < d_{i-1}$ , note that packets labeled  $i$  and  $i - 1$  are both in the queue during the time interval between  $a_i$  and  $d_{i-1}$ . The additional accumulated extra delay due to marked packet transmissions during this time interval is the same for both of these packets, i.e., for  $a_i \leq t \leq d_{i-1}$ , we have,

$$ED_i(t) - ED_i(a_i) = ED_{i-1}(t) - ED_{i-1}(a_i)$$

Thus,

$$ED_i(t) = ED_{i-1}(t) - [ED_{i-1}(a_i) - ED_i(a_i)] \quad (1)$$

Define *ExtraDelayDifference<sub>i</sub>* or *EDD<sub>i</sub>* as  $ED_{i-1}(a_i) - ED_i(a_i)$ , denoting the difference in the extra delay between packet  $i$  and  $(i - 1)$  at time instant  $a_i$ . Now, from (1),

$$ED_i(t) = ED_{i-1}(t) - EDD_i, \quad (2)$$

Therefore,  $ED_i(t)$  can be obtained from  $ED_{i-1}(t)$  and  $EDD_i$ .

For each unmarked packet in the queue awaiting transmission, there is an *EDD* value associated with it. The scheduler keeps these *EDD* values in a separate queue, denoted by *EDDQueue*. The head of this queue, denoted by *EDDHead* contains the *EDD* value of the unmarked head. Similarly, the tail of this queue, denoted by *EDDTail* contains the *EDD* value of the unmarked tail. Let packet  $h$  be the unmarked head at time  $t$ . The next unmarked packet in the queue is packet  $h + 1$ , with a corresponding entry in the *EDD* queue equal to  $EDD_{h+1}$ . Note from Equation (2) that  $ED_{h+1}(t)$  can be calculated from  $EDD_{h+1}$  and the *HeadED* at the time instant when packet  $h$  is transmitted. This quantity now becomes the new *HeadED* after packet  $h$  is transmitted and packet  $h + 1$  is the new unmarked head. *HeadED*, thus, always contains the extra delay corresponding to the current unmarked head.

The value of *EDD* for each unmarked packet is set at the arrival time of the packet. Let packet  $k$  be the unmarked tail at time  $a_{k+1}$  when packet  $k + 1$  arrives. Packet  $k + 1$  at this instant of time will have an extra delay which is the same or different from that of packet  $k$ . Now,  $ED_k(a_{k+1})$  is available in *Tailed* maintained by the scheduler. One needs to find the real extra delay of packet  $k + 1$ , and then set *Tailed* and  $EDD_{k+1}$ . In this paper, we do not describe the details of how we compute

this real extra delay and the complete rationale behind it. The pseudo-code in Fig. 1 and 2, however, describe the  $CL(\alpha)$  algorithm completely. Here, we briefly explain some variables used in the algorithm and which are important to the understanding of the algorithm itself.  $TxTime(P)$  is the transmission time for packet  $P$ .  $UMPD(t)$  is the unmarked packet delay, defined as the time taken to transmit all the unmarked packets which are in the queue at time instant  $t$ . In our algorithm, we assume that unmarked packets cannot pre-empt the transmission of a marked packet. Therefore,  $UMPD(t)$  includes the remaining transmission time of the packet currently being served even if it is a marked packet.

It can be readily verified from the algorithm that  $ED_{k+1}(t)$  is no more than  $ED_k(t)$ . Thus the unmarked head has the largest *ED* compared to other unmarked packets in the queue. When the scheduler tries to send a marked packet, it only needs to make sure that the *HeadED* will not exceed  $\alpha$ . There is no need to check all the *EDs* in the queue.

If *HeadED* will exceed  $\alpha$  upon transmitting a marked packet, the scheduler should drop that marked packet and find an unmarked packet to transmit. Searching the queue for the next unmarked packet is obviously not a scalable option, and will not preserve the  $O(1)$  complexity of this algorithm. Therefore, the  $CL$  scheduler keeps a pointer associated with each element of *EDDQueue* corresponding to the owner of this *EDD* value. From *EDDQueue*, we can find the corresponding unmarked packets. At the time when the scheduler finds out that the marked packet at the head of the queue cannot be transmitted, it will simply look up the pointer associated with the *EDDHead* and send the unmarked packet corresponding to it.

#### IV. ANALYSIS

In this paper, for reasons of brevity, we omit proofs of the following lemmas and theorems on the performance and efficiency of the  $CL(\alpha)$  scheduler. These theorems prove that the  $CL(\alpha)$  scheduler satisfies the requirements laid down in Section II.

**Theorem 1.** The  $CL(\alpha)$  scheduler has a per-packet work complexity of  $O(1)$ .

**Lemma 1.** During the execution of the  $CL(\alpha)$  scheduling discipline, at time instant  $t$  when a new unmarked packet is added into the queue, its extra delay ( $ED(t)$ ) is no more than that of its predecessor, if that predecessor is either waiting in the queue or being transmitted by the scheduler at that time.

**Lemma 2.** During an execution of the  $CL(\alpha)$  scheduler, let  $t$  be the instant of time when an unmarked packet is transmitted. This unmarked packet suffers a delay less than or equal to  $(ED(t) + D_{ref})$ , where  $D_{ref}$  is the delay of the same packet in a reference scheduler which drops all marked packets.

**Theorem 2.** During any execution of the  $CL(\alpha)$  scheduling discipline, the additional delay of an unmarked packet caused by the transmission of marked packets is never greater than  $\alpha$ .

## V. CONCLUDING REMARKS

In this paper, we have defined the requirements of a scheduler serving packets belonging to the controlled load service defined within IETF's Integrated Services architecture. The controlled load service requires source points to regulate the traffic and mark packets that are sent in violation of the traffic contract. One of the requirements we define is that the additional delay of unmarked packets caused due to the transmission of marked packets should be bounded. A  $O(1)$  scheduler to achieve this bound is non-trivial. In this paper, we have proposed the  $CL(\alpha)$  scheduler, which bounds this extra delay to  $\alpha$  or less. The principle used in this algorithm can also be used to schedule flows with multilevel priorities, such as in some real-time video streams as well as in other emerging service models of the Internet that mark packets to identify drop precedences [1,4,5].

## REFERENCES

- [1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services", *IETF Request for Comments 2475*, December 1998.
- [2] J. Wroclawski, "The Use of RSVP with IETF Integrated Services", *IETF Request for Comments 2210*, September 1997.
- [3] J. Wroclawski, "Specification of the Controlled-Load Network Element Service", *IETF Request for Comments 2211*, September 1997.
- [4] D. Clark, "Explicit Allocation of Best-Effort Packet Delivery Service", *IEEE/ACM Transactions of Networking*, vol. 6, no. 4, August 1998.
- [5] W.-C. Cheng, D. D. Kandlur, D. Saha and K. G. Shin, "Adaptive Packet Marking for Maintaining End-to-End Throughput in a Differentiated Services Internet", *IEEE/ACM Transactions on Networking*, Vol. 7, No. 5, October 1999.

## A SIMULATION STUDY OF THE IMPACT OF SWITCHING SYSTEMS ON SELF-SIMILAR PROPERTIES OF TRAFFIC

*Yunkai Zhou and Harish Sethu*

Department of ECE, Drexel University  
3141 Chestnut Street  
Philadelphia, PA 19104-2875.  
{*kenty, sethu*}@ece.drexel.edu

### ABSTRACT

Recent research has shown that traffic in Ethernet and other networks tends to exhibit properties of self-similarity such as long-range dependence and a high degree of correlation between arrivals. This paper investigates the impact of the switching network on the self-similar properties of the traffic. This simulation study reveals that switching networks tend to reduce the self-similarity of highly self-similar traffic. This is because of the truncation of long bursts due to packet discards, and also because of aggregation of flows through concatenated rather than superposed bursts. On the other hand, switching systems have the opposite effect of increasing the self-similarity of input traffic that has no self-similar properties such as traffic with Poisson or uniformly random distributions. This paper also presents simulation-based evidence of the causes behind these phenomena.

### 1. INTRODUCTION

Recent work by many researchers has shown that traffic in Ethernet and other networks tends to be bursty at many or all time-scales [2,6], and that this phenomenon can be mathematically described using the notion of self-similarity. Extensive research has been done on the impact of the self-similar properties of traffic on network design issues, such as queueing performance [10], switch performance [3], congestion control [5] and scheduling algorithms [4]. While it is clear that traffic characteristics have an impact on network design issues, it is also true that the properties of a network have an impact on the characteristics of the traffic as it progresses through the network. Very few studies, however, have addressed this issue of changes in traffic characteristics caused by the network [1, 8, 12, 14]. These studies have focused on only the impact of individual components of a network such as the traffic shaper [12], the packet scheduler [1, 14] or a single-server queue [8], as opposed

to the impact of the entire network as a whole. In addition, studies such as [8] have obtained insightful theoretical results which, however, cannot be readily applied to realistic network environments to solve problems in network engineering. Further, studies such as in [12, 14] only consider short-range burstiness, which does not capture all of the features of self-similar traffic, especially long-range burstiness as observed in [2, 6].

This paper presents a simulation study of the impact of a switching network on the self-similar properties of the traffic, and investigates the causes underlying the observed phenomena. We use self-similar traffic generated using the fractional ARIMA model [7], and a baseline Banyan topology for the switching network. Section 2 discusses the network and the traffic model in greater detail.

Our simulation study reveals that switching networks tend to reduce the self-similarity of highly self-similar traffic. This is because of the truncation of long bursts due to packet discards, and also because of the aggregation of flows through concatenated rather than superposed bursts. On the other hand, switching systems also increase the self-similarity of input traffic that has no self-similar properties such as traffic with Poisson or uniformly random distributions. Section 3 presents these simulation results and the related analysis with simulation-based evidence of the causes behind the phenomena that yield these results. This section also explains our results in relation to those obtained in [8] and [11]. Section 4 concludes the paper.

### 2. NETWORK AND TRAFFIC MODEL

#### 2.1. Network Model

This study uses an  $N \times N$  baseline Banyan multistage network, with  $N$  source nodes and  $N$  destination nodes. The switching network consists of  $\log_m N$  stages of  $m \times m$  switching elements. In Banyan topologies, the path between a source end-point and a destination end-point is unique. This property of Banyan networks helps our study of the

---

This work was supported in part by U.S. Air Force Contract F30602-00-2-0501

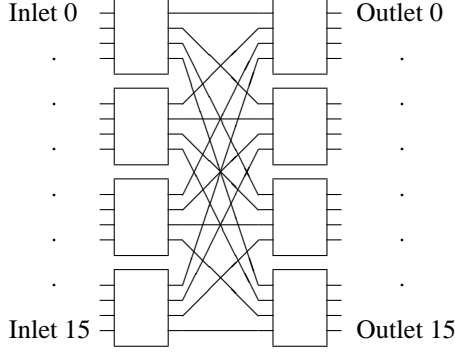


Figure 1: Banyan network with  $N = 16$  and  $m = 4$ .

impact of switching systems, since it eliminates other secondary effects such as due to the choice of a routing algorithm. The popularity of Banyan topologies in real implementations is an additional motivation behind our use of this network model. Figure 1 shows a baseline Banyan network topology with  $N = 16$  and  $m = 4$ .

Each source node consists of  $N$  traffic generators, each of which generates traffic intended for a distinct destination node. Thus, the system consists of a total of  $N^2$  traffic generators. In our simulations, traffic generators are all independent, and generate no more than one packet per cycle. We assume that packet lengths are constant, and that exactly one packet can be transmitted during each cycle across any port. If more than one packet are created in a source node during the same cycle, only one of these is allowed to be transmitted while all the others are buffered in a queue. We assume that the queue sizes at the source nodes are large enough that no packet is ever dropped before it enters the network. Destination nodes drain packets from the output ports of the last stage of switching elements, at the maximum rate of one packet per cycle.

In the switching elements, each input port is associated with an input buffer of a fixed small capacity of 4 packet lengths. Each output port contains a dedicated output buffer. In addition, our simulations also use a shared output buffer of capacity equivalent to 4 packets per output port for additional space for the output queues. Under most traffic conditions, the shared buffer improves performance through better buffer utilization. During each cycle in our simulations, switching elements can accept no more than one packet at each input port into the input queue. Each non-empty output queue transmits exactly one packet to the output port in each cycle. We use the round-robin scheduling algorithm to transfer packets to and from the shared queue. A packet arriving at an input port first enters the associated input buffer, then the shared output buffer, and finally the output buffer corresponding to the destination port. Our model ensures that the maximum bandwidth with which the shared buffer

can be written into or read from, is equal to the maximum aggregate input or output bandwidth of the switch. Packets arriving at a full input buffer are dropped. No packets, however, are dropped at any other point within the switching element, i.e., packets are forwarded to the shared buffer, or to an output buffer only if there is room available.

## 2.2. Traffic Model

We use the fractional autoregressive integrated moving average (FARIMA) model [7] to synthesize self-similar traffic. FARIMA( $p, d, q$ ) is defined as

$$\Phi(B)X_n = \Theta(B)\Delta^{-d}\epsilon_n,$$

where  $B$  is the backward operator, i.e.,  $Bx_n = x_{n-1}$ . The definition above can be also expressed as

$$X_i = \Delta^{-d}\epsilon_i - \theta_1\Delta^{-d}\epsilon_{i-1} - \dots - \theta_q\Delta^{-d}\epsilon_{i-q} + \phi_1X_{i-1} + \dots + \phi_pX_{i-p}. \quad (1)$$

In equation (1),  $\Delta^{-d}$  is defined as  $\Delta^{-d} = \sum_{i=0}^{\infty} b_i(-d)B^i$ , where  $b_0(-d) = 1$  and

$$b_i(-d) = \frac{\Gamma(i+d)}{\Gamma(d)\Gamma(i+1)}, i = 1, 2, \dots$$

When the innovation  $\epsilon_i$  is a stable process with index  $\alpha$ , i.e.,  $\epsilon_i \sim S_\alpha(\sigma, \beta, \mu)$ , the Hurst parameter,  $H$ , and the quantities  $\alpha$  and  $d$  are related by  $d = H - 1/\alpha$ . In this paper, we use the Hurst parameter as the measure of the degree of self-similarity. The Hurst parameter has a range of  $0.5 \leq H \leq 1$ , and a larger value of  $H$  implies a higher degree of self-similarity. Throughout our work, we use  $\alpha = 1.2, \sigma = 1, \beta = 0, \mu = 0, p = 50$ , and  $q = 400$ .  $\Theta(B)$  is generated by selecting  $\theta_i$  in  $[0, 0.05]$  randomly and independently. Unlike  $\Theta(B)$ ,  $\Phi(B)$  is generated by selecting  $p/2$  complex roots and their conjugates, since  $X_i$  converges only if all roots of  $\Phi(B)$  are in the unit circle. The real and imaginary components of each root are uniform in  $[0, 0.05]$ . Finally, we normalize  $X_i$  to a series of 1's or 0's indicating whether or not a packet is generated during a given cycle.

The variance-time plot [9] is used to estimate the Hurst parameter of observed network traffic. For a self-similar time series  $X(k)$ ,  $X_m(k)$  is defined as

$$X_m(k) = \frac{1}{m} \sum_{i=mk}^{(m+1)k-1} X(i),$$

and

$$\text{var}X_m = \text{var}X/m^\beta,$$

where  $H = 1 - \beta/2$ . Taking the logarithm of the equation above, we get,

$$\log \text{var}X_m = -\beta \log m + \log \text{var}X,$$

From the above equation, the Hurst parameter is determined by the slope of the plot of  $\log \text{var}X_m$  vs.  $\log m$ .



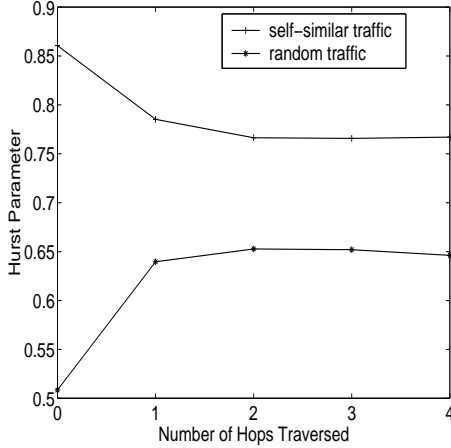


Figure 2: Per-hop changes in self-similarity.

### 3. SIMULATION RESULTS AND ANALYSIS

In a switching element with buffers, a flow typically consumes more space during a bursty period. Under such conditions, depending on the buffer sharing policy and the buffer sizes, either other flows suffer from less empty space, or the bursty flow suffers a higher packet loss rate. In either of these cases, the traffic characteristics change due to delays or losses or both. If two or more flows are bursty at the same time, these effects are further magnified. This section presents our study of these effects on the self-similar properties of traffic.

Our study includes two kinds of traffic sources, self-similar and uniform random traffic. A uniform random traffic source generates a packet during each cycle with a certain probability  $p$ , with uniformly distributed packet destinations. Like Poisson traffic, uniform random traffic has a Hurst parameter of 0.5, indicating that it has no self-similar properties.

Our simulation study shows that the impact of a switching system on the self-similarity of the traffic depends on the self-similarity of the input traffic itself. A switching system reduces the self-similarity of highly self-similar traffic, while it increases that of non-self-similar traffic such as uniform random traffic. Figure 2 illustrates this phenomenon of the opposite nature of the effects observed depending on the self-similarity of the input traffic itself. When the traffic is uniformly random, the Hurst parameter increases from 0.5 to 0.64 after the first stage and stays around 0.65 thereafter. When the input traffic has a high level of self-similarity, the Hurst parameter drops from 0.86 to 0.78 after the first stage, and further to 0.76 after the second stage. This interesting phenomenon shows us that, switching networks have the effect of shaping the traffic characteristics to a moderate level of self-similarity. In the following, we investigate

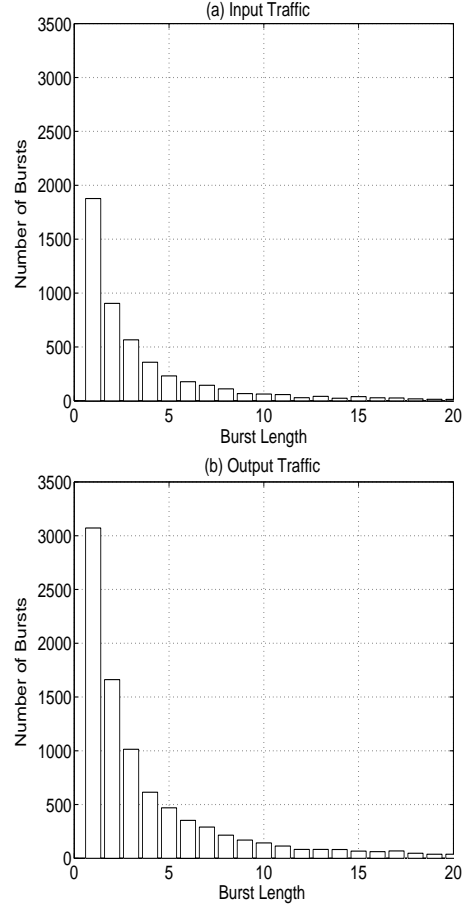


Figure 3: Distribution of short burst length, (a) input traffic and (b) output traffic.

and present simulation-based evidence of the causes of this phenomenon.

In the case of uniform random traffic, the probability of packet arrivals during each cycle is independent of the packet arrival pattern during the previous cycles. As the traffic progresses through a switching network with buffers in the switching elements, this independence assumption progressively becomes less valid. Packets for the same output port that independently arrive at different times, due to congestion, end up waiting in the buffers for transmission, and get transmitted in a burst at the output port. This phenomenon adds burstiness to the traffic at each new hop in the path of the traffic, changing the output traffic characteristics to something other than random uniform traffic. Packet arrivals at subsequent hops of the network are now correlated, as reflected in the increased Hurst parameter of the traffic.

The distribution of burst lengths in highly self-similar traffic is heavy-tailed, i.e., the probability distribution is given by  $P[X > x] \sim x^{-\alpha}$ . Such a distribution decays

Number of Nodes	$H$ (Input Traffic)	$H$ (Output Traffic)	Percentage Decrease
$2 \times 2$	0.860	0.864	$\sim 0$
$4 \times 4$	0.859	0.843	2%
$8 \times 8$	0.863	0.812	6%
$16 \times 16$	0.861	0.763	11%

Table 1: Self-similarity of traffic vs. number of nodes.

more slowly than exponentially, causing a high likelihood of long bursts in self-similar traffic. However, because of congestion and limited buffering capacities in the switching elements, long bursts do not easily survive in switching networks. In fact, long bursts self-destruct through causing congestion, triggering discarding of packets and thus breaking the long burst into smaller ones. For example, in our simulations, the longest burst observed at the traffic source had more than 6,000 consecutive packets, while the output traffic had no bursts longer than 900 packets. This is also illustrated in Figure 3, which shows that the distribution of short burst lengths of the input and the output traffic of the network. The output traffic has a larger percentage of shorter bursts, and with a significantly smaller average burst length. Note that the relative increase in shorter bursts increases the value of the index  $\alpha$  in a heavy-tailed distribution given by  $P[X > x] \sim x^{-\alpha}$ . This, in turn, has the effect of reducing the Hurst parameter since, as shown in [13],  $H = (3 - \alpha)/2$ .

In addition to the reduction in burst lengths, the other reason for this phenomenon is that aggregation of flows in networks typically has the effect of reducing variation over larger scales. It should be understood that this phenomenon is quite different from that shown in [11]. Willinger *et al.* show that a superposition of many *ON/OFF* traffic sources exhibits properties of self-similarity when the lengths of the *ON* and *OFF* periods are independent and follow a heavy-tailed distribution. Because of the limited bandwidth of output links, a true superposition is never possible in switching networks. Bursts are actually concatenated rather than superposed on top of each other on the output links. A superposition increases variation across scales, but a concatenation actually has the effect of spreading out the peaks and thus smoothening out the variations.

The phenomenon discussed above can be verified through simulation using a single  $m \times m$  switching element and varying  $m$ . In this model, each output link is fed by  $m$  self-similar traffic sources at the inputs. Table 1 shows the impact on the self-similarity of the output traffic for different values of  $m$ . As can be observed from Table 1, the self-similarity of traffic decreases as the level of aggregation increases. This reduction in the self-similarity of traffic with aggregation, is also the reason that highly self-similar traffic

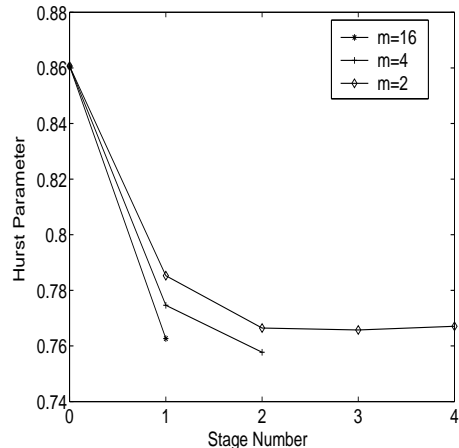


Figure 4: Per-hop changes for different switch sizes.

reduces in self-similarity as it progresses through each hop in the switching network. This is easily understood from noting that the output links further hops away from the traffic sources carry more of an aggregated traffic than the ones closer to the sources.

The same phenomenon is apparent in the impact of the size of switching elements used in the topology of a switching network. A network designed using  $2 \times 2$  switching elements, as compared to  $4 \times 4$  switching elements, will contribute to a smaller decrease in the observed Hurst parameter after the first hop. A network with  $4 \times 4$  switching elements achieves the same level of aggregation in fewer hops than one using  $2 \times 2$  switching elements. Figure 4 shows the per-hop changes in the self-similarity of traffic for switching networks with different sizes of switching elements.

It is worthwhile to discuss our results in relation to those obtained by Song *et al.* [8] in their study of self-similarity of output traffic at a single server with an infinite buffer. It is proved in [8] that if the queue length has finite variance, the self-similar properties of input and output traffic remain the same. In fact, it is shown that both the input and output traffic have the same Hurst parameter. Noting that it is unrealistic to assume an infinite buffer, the authors in [8] argue that in real switches, the condition that queue length has finite variance is always satisfied. However, another important impact of finite buffers should be considered—traffic can be accepted into the buffer only if there is available space. In the absence of a feedback mechanism, packet discarding becomes inevitable which changes the traffic characteristics; in the presence of a feedback mechanism such as credit-based flow control, the characteristics of arriving traffic itself changes. A second important reason for the apparent discrepancy between our results and that in [8] is that our results use the self-similar properties of *aggregate* traffic at each output link, while the results in [8] compare

the properties of *individual* flows before and after service by the server. Our approach to only analyze aggregate traffic is motivated by the fact that most switches and routers do not maintain per-flow queueing, and therefore, only the characteristics of the aggregate traffic at each input or output link is important to the performance and related issues in the design of switches and routers.

All of the results presented in this paper were obtained at moderate or heavy traffic loads. As one might expect, the impact of the switching network on the traffic characteristics is minimal when the traffic load is small.

#### 4. CONCLUSION

In this paper, we have presented simulation studies that show that highly self-similar input traffic reduces in its self-similarity as it progresses through a switching network. Our analysis indicates that this phenomenon is caused by the truncation of long bursts due to packet discarding, and by the aggregation of flows through concatenation of bursts. On the other hand, during periods of congestion in networks with buffers, input traffic with no self-similar properties increases in self-similarity as it progresses through the network.

These results have important implications relevant to the design of routers and switches. For example, our results suggest that core Internet routers receive traffic that is much less self-similar than traffic that emerges out of border routers directly connected to Ethernet LANs.

#### Acknowledgments

The simulation code to generate the self-similar traffic was partly written by Haiguang Cheng. The authors would also like to gratefully acknowledge his contribution through discussions that further improved the traffic model.

#### REFERENCES

- [1] S. Borst, O. Boxma and P. Jelenković. "Asymptotic Behavior of Generalized Processor Sharing with Long-Tailed Traffic Sources". *Proceedings of IEEE INFOCOMM*, Tel Aviv, Israel, Mar. 2000.
- [2] M. E. Crovella and A. Bestavros. "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes". *IEEE/ACM Transactions on Networking*, vol 5, no. 6, Dec. 1997.
- [3] S. Fong and S. Singh. "Performance Evaluation of Shared-Buffer ATM Switches Under Self-Similar Traffic". *Proceedings of IEEE International Performance, Computing, and Communications Conference*, Arizona, 1997.
- [4] R. G. Garroppo, S. Giordano, S. Miduri, M. Pagano and F. Russo. "Statistical Multiplexing of Self-Similar VBR Video-conferencing Traffic". *Proceedings of GLOBECOM*, 1997.
- [5] A. Gersht, G. Pathak and A. Shulman. "Burst Level Congestion Control in ATM Networks". *Proceedings of IEEE Symposium on Computer and Communications*, Athens, Greece, July 1998.
- [6] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson. "On the Self-Similar Nature of Ethernet Traffic (Extended Version)". *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, Feb. 1994.
- [7] G. Samorodnitsky and M. S. Taqqu. *Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance*. Chapman & Hall, NY, 1994.
- [8] S. Song, J. K.-Y. Ng and B. Tang. "On the Self-Similarity Property of the Output Process from a Network Server with Self-Similar Input Traffic". *Proceedings of Sixth International Conference on Real-Time Computing Systems and Applications*, pp. 128–132, Dec. 1999.
- [9] W. Stallings. *High-Speed Networks: TCP/IP and ATM Design Principles*. Prentice Hall, Upper Saddle River, NJ. 1998.
- [10] B. Tsybakov and N. D. Georganas. "Overflow Probability in an ATM Queue with Self-Similar Input Traffic". *Proceedings of IEEE International Conference on Communications*, Montreal, Canada, June 1997.
- [11] W. Willinger, M. S. Taqqu, R. Sherman and D. V. Wilson. "Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level". *IEEE/ACM Transactions on Networking*, vol.5, no.1, Feb. 1997.
- [12] S. Wittevrongel and H. Bruneel. "Output Traffic Analysis of a Leaky Bucket Traffic Shaper Fed by a Bursty Source". *Proceedings of IEEE International Conference on Communications*, pp. 1581–1585, 1994.
- [13] X. Yang, A. P. Petropulu and V. Adams. "The Extended ON/OFF Model for High-Speed Data Network". *Workshop on Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics*, Washington, D.C., Jun. 1999.
- [14] H. Zhang. "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks." *Proceedings of the IEEE*, vol. 83, no. 10, Oct. 1995.

PIGLET: AN OPERATING SYSTEM FOR NETWORK  
APPLIANCES

STEVE J. MUIR

A DISSERTATION

in

COMPUTER AND INFORMATION SCIENCE

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the  
Requirements for the Degree of Doctor of Philosophy

2001

---

Jonathan M. Smith  
Supervisor of Dissertation

---

Val Tannen  
Graduate Group Chair

© Copyright 2001

by

Steve J. Muir

## ACKNOWLEDGEMENTS

First and foremost, I would like to thank *Jonathan Smith* for his guidance and advice over the past four and a half years. His knowledge, experience and unfailing enthusiasm for this project inspired me to persevere.

My thesis committee—*Michael Greenwald, Dave Farber, Roch Guerin, Derek Mcauley and Larry Peterson*—humoured my initial optimistic estimates of how much work was to be done but their patience was greatly appreciated. Their combined wisdom and insightful comments improved this thesis no end and provided ample food for future thought.

*Luke Hornof* kept me sane, enlightened and amused, *Sanjay Udani* proved to be a worthy *Scrabble* adversary, and *Dave Azari* taught me to shoot hoops. *Geoff Milord* did an excellent job of implementing various Piglet hacks and making me look organised.

*Paul Beavan* at IBM Hursley gave me the opportunity to work on a substantial project before I knew the first thing about Computer Science: from that small seed everything else has grown.

*Aled Edwards* and everyone else I worked with at HP Labs Bristol introduced me to the research environment, got me hooked and put the beginnings of this thesis into my head. Aled deserves special mention for teaching so much by his example.

My internships at DEC/Compaq's Systems Research Center and NEC USA's Advanced Development Center helped broaden my horizons. Thanks to *Sanjay Ghemawat* and *Zoran Miljanic* for giving me those opportunities.

My time at Penn was made enjoyable and educational by all my fellow students in the Distributed Systems Lab. Particular thanks go out to everybody who shared an office with me and had to put up with the noise and clutter.

*Vanu Bose* and my colleagues at *Vanu, Inc.* have inherited the noise but given me the chance to work in an exciting new venture.

*Katherine Becker* coaxed me along when I was least motivated to sit down and write.

To all my friends, wherever you are: I couldn't ask for better ones.

Last, but certainly not least, thanks go to my family for always encouraging me to pursue whatever goals I set my sights on.

This work was supported by DARPA under Contract #N66001-96-C-852 and Cooperative Agreement # F30602-00-2-0501, and by the US National Science Foundation under Grants #ANI98-13875 and ANI 99-06855.

## ABSTRACT

### PIGLET: AN OPERATING SYSTEM FOR NETWORK APPLIANCES

Steve J. Muir

Jonathan M. Smith

Advances in the performance of commodity hardware and acceptance of open-source software have recently led to the increased use of systems based upon a combination thereof as network appliances. The principal thesis of this dissertation is that such appliances can operate more efficiently if their operating system is designed specifically for that task. I present *Piglet* as a novel design for such an operating system, and describe its implementation and evaluation in such a context.

The core of the Piglet architecture is the *Active Kernel*. A dedicated kernel processor provides concurrency between kernel and applications and enables asynchronous shared-memory communication. The use of shared objects for all communication between applications and the kernel permits Piglet to offer much more efficient mechanisms for the invocation of system services than a conventional kernel.

The experimental results presented herein demonstrate how these fundamental kernel features make Piglet a more efficient operating system for network appliances: reduced latency for individual operations and increased aggregate data throughput. In addition to detailing microbenchmark results to support those claims, I show how an existing web server application can readily make use of those enhancements to achieve a performance boost over standard operating systems.

Finally, this dissertation examines, both quantitatively and qualitatively, some of the challenges and problems presented by the implementation of the Piglet architecture, and proposes solutions and/or architectural changes to address those concerns.



# Contents

<b>Acknowledgements</b>	<b>iii</b>
<b>Abstract</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 What is a Network Appliance? . . . . .	3
1.2 Why a New Operating System Architecture? . . . . .	3
1.3 Contributions of this Dissertation . . . . .	4
1.4 Outline of the Dissertation . . . . .	5
<b>2 Intrusion—When Operating Systems Go Bad</b>	<b>6</b>
2.1 Resource Intrusion . . . . .	7
2.2 Policy Intrusion—Common-Case Optimisation Considered Harmful . . . . .	8
2.3 Mechanism Intrusion—the Price of Safety . . . . .	9
2.3.1 Synchronous Intrusion—Traps . . . . .	11
2.3.2 Asynchronous Intrusion—Interrupts . . . . .	11
2.4 Correlated vs. Uncorrelated Intrusion . . . . .	12
2.5 Intrusion in Multiprocessor Systems . . . . .	12
2.6 Intrusion Factor—an Intrusion Metric . . . . .	13
2.6.1 Livelock—the Curse of Infinite Intrusion Factor . . . . .	14
2.7 Evaluating the Cost of Intrusion . . . . .	14
<b>3 The Piglet Architecture</b>	<b>17</b>
3.1 The Evolution of Operating Systems . . . . .	17

3.1.1	The Microkernel . . . . .	19
3.1.2	Vertically-Structured Systems . . . . .	21
3.1.3	The Next Step—an Active Kernel . . . . .	22
3.2	Anatomy of Piglet . . . . .	24
3.2.1	Primary Features of Piglet . . . . .	24
3.2.2	Secondary Architectural Features . . . . .	25
3.3	A Dedicated Kernel Processor . . . . .	26
3.3.1	Client Page Monitor . . . . .	27
3.3.2	Integrated Service Scheduler . . . . .	27
3.3.3	Host Communication Module . . . . .	29
3.3.4	Scheduling in the Polling Function . . . . .	29
3.4	No Asynchronous Interrupts . . . . .	30
3.5	Shared-Memory Communication . . . . .	32
3.6	The Piglet-Application Interface . . . . .	36
3.6.1	API Basics . . . . .	37
3.6.2	Asynchronicity Creates Concurrency . . . . .	38
3.6.3	References Arguments and Speculation . . . . .	39
3.6.4	Optimistic Service Invocation . . . . .	39
<b>4</b>	<b>Implementing Piglet</b>	<b>41</b>
4.1	Building the Linux/Piglet Hybrid . . . . .	41
4.1.1	Linux Kernel Hooks . . . . .	43
4.1.2	Using Host Services Within Piglet . . . . .	44
4.1.3	The Host Communications Module . . . . .	45
4.2	The Piglet Network Subsystem . . . . .	46
4.2.1	Frame Transmission . . . . .	51
4.2.2	Frame Reception . . . . .	52
4.2.3	Frameset Creation . . . . .	54
<b>5</b>	<b>Evaluating Piglet</b>	<b>56</b>
5.1	The Overhead of Service Invocation . . . . .	57

5.1.1	Measurement of Round-Trip Latency . . . . .	57
5.1.2	Analysis of System-Call Overheads . . . . .	60
5.2	Concurrent Data Throughput . . . . .	63
5.2.1	Application Throughput Analysis . . . . .	65
5.3	Profiling the Polling Function . . . . .	67
5.3.1	PSR Processing Time . . . . .	67
5.3.2	Frameset Polling Overhead . . . . .	68
5.3.3	Frame Transmission Cost . . . . .	69
5.3.4	Device Polling Overhead . . . . .	70
5.3.5	CPU Utilisation . . . . .	71
5.3.6	Analysis of Polling Function . . . . .	72
<b>6</b>	<b>Application-Level Performance: The Flash Web Server</b>	<b>74</b>
6.1	Porting Flash to Piglet . . . . .	74
6.2	Evaluating Flash's Performance . . . . .	77
<b>7</b>	<b>Related Work</b>	<b>80</b>
7.1	Intrusion . . . . .	80
7.1.1	Policy Intrusion . . . . .	81
7.1.2	Mechanism Intrusion . . . . .	82
7.1.3	Multiprocessor Intrusion . . . . .	83
7.2	Alternative Operating System Architectures . . . . .	84
7.2.1	Microkernels . . . . .	85
7.2.2	Vertically-Structured Operating Systems . . . . .	87
7.2.3	Extensible Operating Systems . . . . .	88
7.3	Operating Systems for Network Appliances . . . . .	89
7.4	Message-based Systems . . . . .	90
<b>8</b>	<b>Conclusions and Future Research</b>	<b>92</b>
8.1	Analysing Operating System Intrusion . . . . .	93
8.2	Piglet—Reducing Intrusion by Activating the Kernel . . . . .	93

8.3	Performance of the Active Kernel . . . . .	94
8.4	Future Work . . . . .	94
8.4.1	Optimising the Active Kernel . . . . .	94
8.4.2	Evaluation of Alternative Applications . . . . .	95
<b>A</b>	<b>Linux ping Execution Trace</b>	<b>96</b>

# List of Tables

2.1	Measured time costs of OS intrusion . . . . .	15
2.2	Comparative costs of system calls . . . . .	15
3.1	Network-related services in the Piglet kernel . . . . .	28
5.1	Mean round-trip times . . . . .	59
5.2	Mean processing time for various PSRs . . . . .	67
5.3	Total frameset processing time for $n$ pending frames . . . . .	69
5.4	Device event handling time in different states . . . . .	70
5.5	CPU utilisation and polling period . . . . .	71
6.1	Distribution of file sizes for Webstone benchmark . . . . .	77
6.2	Flash performance for the Webstone benchmark . . . . .	77
6.3	Flash performance for the <i>curl</i> test . . . . .	78

# List of Figures

3.1	Monolithic operating system structure . . . . .	18
3.2	Microkernel-based operating system structure . . . . .	20
3.3	Vertically-structured operating system . . . . .	21
3.4	Active kernel-based operating system . . . . .	22
3.5	Structure of the <i>main</i> function . . . . .	27
3.6	Basic structure of the Piglet shared queue object . . . . .	34
3.7	Structure of the Piglet Posted Service Request . . . . .	36
4.1	Hybrid Linux/Piglet kernel structure . . . . .	42
4.2	Network module structure . . . . .	47
4.3	Frameset and user-level frame header structure . . . . .	48
4.4	Frame header state transitions . . . . .	50
4.5	Frameset creation: an example of chained posted service requests . . . . .	54
5.1	Command used to measure round-trip latencies. . . . .	58
5.2	Distribution of round-trip times as a function of payload size . . . . .	58
5.3	Linux <b>ping</b> trace . . . . .	61
5.4	Piglet <b>ping</b> trace . . . . .	61
5.5	Testbed configuration used for bandwidth measurement . . . . .	63
5.6	Mean aggregate throughput for $n$ concurrent TCP connections . . . . .	64
5.7	Frameset polling time as a function of number of clients . . . . .	68
6.1	Flash performance for $n$ server processes . . . . .	79

# Chapter 1

## Introduction

The Internet is here—it is ubiquitous, it is diverse, and it is growing. Ubiquitous: any two computing devices anywhere in the world can communicate with each other across it. Diverse: the number and variety of connected devices ranges from the tiniest cellular phones up to the largest supercomputers, sending many different types of data across a single physical network. It's growing: Kahn and Cerf [40] estimated in December 1999 that the Internet would connect 300 million computers by the end of 2000, and continue growing at 80% per year.

Compare the success of the Internet with other major communications networks, even just in the United States. The plain old telephone system (POTS) is certainly ubiquitous and works extremely well, but one can hardly claim diversity of end systems or traffic types as one of its successes. The same report by Kahn and Cerf has the telephone network growing at 5–10%, at which rate it will be surpassed in size by the Internet around the end of 2006. Wireless communications networks are growing, but competing standards have thus far prevented ubiquity and only recently have networks diversified beyond voice calls between mobile phones.

A large part of the success of the Internet stems from the use of the same computer systems within the network as are connected to the network. Familiarity with these systems by network users enabled the easy and rapid deployment of services within the network. Even as far back as the ARPANET, data networks have used the same type of systems as the network nodes as also constituted the network leafs; many of the original IP gateways,

themselves various *LSI-11* machines, primarily routed messages between similar systems used by the connected research institutions. The widespread adoption of the *Internet Protocol (IP)* [71] allowed many different hardware systems to all connect to the same network.

This homogeneity of computing platform, combined with the tremendous prevalence of the UNIX operating system within computer science research institutions, meant that soon UNIX not only ran on most computers connected to the network but also the majority within it. When the need to deploy network services arose the choice of platform for those services was obvious; hence the almost universal provision of network services by UNIX-based systems e.g., the *domain name service(DNS)* [56, 57] and the *simple mail transfer protocol (SMTP)* [72] being commonly implemented by the UNIX programs *bind* [37] and *sendmail* [79] respectively.

With the advent of the World Wide Web in the early 1990s two new classes of network service have become predominant—the Web server and Web cache. While these services are also often provided by UNIX-based systems, albeit running less ubiquitous programs (*Apache* [1] and *Squid* [3] respectively), there has recently been increased interest in having them provided by systems designed exclusively for and dedicated to that task—server appliances. Similarly, after years of moving toward dedicated hardware-based forwarding, router vendors are starting to return to general-purpose platforms as the basis for their systems: *Microprocessor Report* reports [25] that Cisco is using a network processor rather than fixed-function ASIC because the latter lacks the flexibility required to handle new protocols.

While the predominance of a single, general-purpose operating system has led to much of the success of the Internet, it is the claim of this thesis that new operating systems designed specifically for *network appliances* are necessary to continue that success in the 21st century.



## 1.1 What is a Network Appliance?

**Definition:** a network appliance is any system with the primary and sole task of enabling the provision of network services to client systems.

The term ‘*client system*’ is used in no stronger sense than to mean the recipient of data from another computer system—in particular, it does not imply a particular communication architecture. Appliances are considered to be systems which are specialised, although not necessarily at the hardware level, for a specific task, thus systems which enable provision of network services as one of their many roles are excluded from this categorisation.

Network appliances can be further divided into two classes: those systems which provide network services, or equivalently, the data which constitutes those services; and those systems which convey that data from the provider to the client. These two classes are denoted *server appliances* and *forwarding appliances* respectively—the terms *server* and *router* are roughly equivalent but are avoided here due to additional connotations associated with them.

Both server and forwarding appliances may be implemented using either special- or general-purpose hardware platforms. However, increases in the last ten years in the computing power of commodity computer systems, specifically Intel *x86*/IBM PC-compatible machines, and a related but lesser increase in I/O throughput, have made such systems the platform of choice for network appliances.

## 1.2 Why a New Operating System Architecture?

The motivation to design a new operating system arose from two independent factors: deficiencies in the use of existing, general-purpose operating systems for network appliances, and the increasing availability of multiprocessor variants of the commodity PC. This thesis demonstrates how an appropriately designed operating system leverages the benefits of the multiprocessor platform to address the problems of the appliance operating system.

The chief deficiency arising from the use of general-purpose operating systems as the

basis for a network appliance is the failure to take advantage of favourable workload characteristics. More specifically, network appliances primarily move data rather than create it, either from storage to network—server appliances—or network to network—forwarding appliances. While this characterisation is obviously true for the majority of network appliances e.g., web caches and proxies, email servers, firewalls, routers, etc., there is one important special case to consider in more detail: web servers providing dynamic web content. Dynamic web pages have undoubtedly become a significant fraction of web traffic, but many of the most common dynamic page technologies, such as JavaScript and Java applets, are implemented purely by the client (browser). In addition, at least one study of server workloads [46] found that server-side dynamic content (CGI scripts in that study) accounted for a small fraction, typically less than 10%, of total workload.

Thus one may indeed characterise the workload of network appliances as being predominantly data transfer rather than computation, and hence the operations performed by software will obviously be different than those of a system used for more general computational tasks. It therefore appears reasonable to assume that an operating system specialised to the task of data movement may make a more efficient basis for a network appliance than a more general system.

Similarly, an operating system which is designed specifically to run on a multiprocessor system may be able to better utilise multiple processors than a uniprocessor operating system retrofitted with the requisite facilities e.g., concurrency control, scheduling. Combine this with the benefits of optimisation for the network appliance workload and significant benefits may be expected.

### 1.3 Contributions of this Dissertation

The primary goal of this work was the design, implementation, testing and evaluation of an operating system targeted specifically at the network appliance application domain and to be deployed exclusively on commodity multiprocessor hardware. As steps toward this goal, and as the ultimate product of this work, this dissertation makes several contributions to the fields of operating systems and network appliances:

- A characterisation of the overheads imposed by existing operating systems upon application programs.
- An architectural design for a multiprocessor operating system specialised to the task of hosting a network appliance system.
- An implementation of this architecture using commodity hardware and existing operating systems to provide a convenient development and evaluation platform.
- Analysis of this prototype implementation, including both latency- and throughput-based microbenchmark results.
- Porting of an existing server appliance application program—the *Flash* [69] web server—to the prototype operating system.
- Measurement of the application-level performance benefits compared to a conventional, general-purpose operating system.

## 1.4 Outline of the Dissertation

After this brief introduction to the goals of this dissertation I begin with a qualitative analysis of the costs incurred by application programs due to the structure of existing operating systems. In Chapters 3 and 4 a design for an architecture which reduces these costs is introduced, and a prototype implementation described.

This implementation is then evaluated at the microbenchmark level using a number of experiments, described in Chapter 5. The following chapter describes the porting of a particular application, the *Flash* web server, to the Piglet prototype and provides a comparison of application performance on Piglet and Linux platforms.

The dissertation concludes by comparing the operating system proposed here with other work which addresses the same concerns. The contributions of this thesis and topics of future work are investigated in Chapter 8.

## Chapter 2

# Intrusion—When Operating Systems Go Bad

The primary role of a modern operating system is enabling the concurrent execution of multiple independent programs. This is equivalent to the multiplexing of shared resources among those programs—a task usually accomplished by the virtualisation of those resources, consisting of the following interrelated functions:

1. *Protection*—preventing applications from accessing resources in modes other than which they have been specifically authorised to do so.
2. *Multiplexing*—providing multiple applications with independent virtualisations of a single physical resource.
3. *Translation*—mapping each application’s virtualisation onto the underlying physical resource(s).

The composition of these *virtualisation primitives* allows an operating system to share physical resources among the independent programs executing at any given time. As an example, consider virtual memory: page tables provide translations from virtual address spaces onto the multiplexed physical memory and are protected from unauthorised access by application programs.

Unfortunately these virtualisation primitives have often been needlessly conflated with the higher-level functions provided by operating systems, such as filesystems, network protocol stacks, and execution of binary programs. This conflation arises from what was historically an important role of an operating system: to provide a library of common functions which could be used by the single program executing at a given time on the host computer system [50].

This intimate coupling between virtualisation and higher-level functionality unfortunately leads to non-optimal application behaviour (see Section 7.1.1). Hence the recent interest in operating systems which decouple virtualisation from higher-level functionality, such as the University of Cambridge’s *Nemesis* kernel, and MIT’s *Exokernel*.

All forms of intrusion reduce the availability of physical resources to applications—I have chosen, however, to separate them into two categories: *resource intrusion* and *time intrusion*. The latter is further subdivided by the basis of the intrusion: either caused by the policies implemented by the operating system—*policy intrusion*—or due to the mechanisms employed by the operating system—*mechanism intrusion*.

A rule of thumb for determining whether resource usage by an operating system should be considered intrusion is whether an application (or group of applications) running with exclusive, direct access to all physical resources would exhibit the same usage.

## 2.1 Resource Intrusion

Resource intrusion denotes the use of physical resources e.g., memory, disk blocks, network bandwidth, by an operating system for its own purposes, or equivalently for purposes which do not further the progress of an application. As an example, memory used by the operating system to store information about each running process, or about the use of memory pages, is obviously not available for an application’s own purposes. Similarly, disk blocks used to store metadata e.g., inodes in a UNIX filesystem, and network headers used to multiplex multiple data streams onto a physical network also both constitute operating system (at least for accounting purposes in the networking case) intrusion.

While resource intrusion is both important and fertile ground for research it is not

covered further in this thesis. This is primarily to limit the scope of the work, but it may also be observed that in today’s resource-rich computing environments the small overheads incurred due to the operating system are not worth consideration. However, that is not to say that such work is not relevant or has not been done e.g., Van Jacobson’s TCP header compression [38], the *ReiserFS* [78] tree-structured file system.

## 2.2 Policy Intrusion—Common-Case Optimisation Considered Harmful

*Policy Intrusion* arises when the functions provided by an operating system embody policies which have detrimental effects upon application behaviour. A classic example of this effect was observed in Stonebraker’s seminal paper on “*Operating System Support for Database Management*” [85], where the policies implemented by a general-purpose operating system were determined to be non-optimal for the specific application of database management. Policy intrusion generally arises not from an operating system’s resource management policies being optimised for the most common (general) case, but from lack of provisions for circumventing these policy decisions.

Examples of such policy decisions include file prefetching and network buffer overflow. File prefetching is performed by an operating system to increase the efficiency of accesses to storage devices—operating systems often requests multiple blocks at once to amortise the overhead associated with requesting a block of data over multiple blocks. The file prefetching policy determines which additional blocks should be requested if less than the multiple number are specified by the application; the most common such policy is sequential lookahead, where the operating system assumes that if block  $n$  is requested by an application then blocks  $n + 1, n + 2, \dots$  will also be requested in the near future. While this may be true for general-purpose applications which tend to read files sequentially it is unfortunately completely inappropriate for (say) database management systems which access files randomly. The operating system thus wastes bandwidth and time processing blocks which will never be needed. Similar inefficiency may also occur when a CPU fetches a whole cache line from memory to satisfy an access to only a single word within that

line e.g., non-sequential access to large arrays; this is an example of CPU policy intrusion rather than operating system intrusion however.

Network buffer overflow occurs when a network packet arrives destined for an application which already has full buffers. Since there is no space for the new packet the operating system must have a policy for what action to take in such circumstances. Although the common policy of discarding the new packet is acceptable for most applications, for an important class of program, namely those which handle time-dependent or idempotent data, it would be preferable to discard the oldest data to create room for the new.

Thus different projects to reduce policy intrusion have taken different tacks toward the same goal. One approach, exemplified by those systems known as *vertical operating systems* or *exokernels*, removes as much policy as possible from the operating system kernel, instead relying upon applications and libraries to implement all higher-level resource management with the appropriate policies. An alternative approach, taken by projects such as Carnegie-Mellon University's *TIP* system [10, 70] allows applications to specify *hints* to the operating system about the preferred resource management strategy for that application. Similarly, the `vadvise` system call is provided in some versions of BSD UNIX to allow applications to 'advise' the kernel of particular virtual memory behaviour patterns.

## 2.3 Mechanism Intrusion—the Price of Safety

In order to support the first primitive of resource virtualisation i.e., protection, an operating system must isolate physical resources from the applications which use them. Otherwise a malicious application with direct access to the resource, say through its I/O control ports, can subvert whatever multiplexing the operating system performs, either to acquire more resources for itself, or to interfere with the resources allotted to another application.

Protection is almost universally implemented by partitioning the program execution domain into privileged and unprivileged regions, which are conventionally referred to as *supervisor* and *user* levels. The operating system runs at the supervisor level, with full access to all CPU instructions and capabilities, while applications run at user level, able to access only a 'safe' subset of the instruction set.

However, it is worth noting that privilege levels are not the only means of preventing unauthorised access to a physical resource: language-based protection and naming have also been used to do so. Examples of systems which derived protection from a high-level language include the *LISP machine* [23], *Swift* [12], based upon the *CLU* language [44], and the Java-based *KaffeOS* [4]. The *Anonymous RPC* [91] system eschews privilege-based protection in favour of a probabilistic approach: by randomly distributing virtual pages within a large, sparse address space and making the penalty for an invalid access sufficiently high it becomes infeasible for applications to scan the address space hoping to stumble across other applications' memory pages. *Amoeba* [86] uses cryptographic mechanisms to enforce protection of capabilities granted to applications—capabilities are encoded in such a manner that applications cannot make modifications to them without violating integrity checks included in the capability. Such schemes are not without cost of their own though, nor can they be applied to all resources, so they should not be considered a perfect solution.

Once the operating system kernel has erected privilege boundaries around itself to provide resource protection it becomes necessary to cross those boundaries when kernel services must be invoked. These privilege boundary crossings occur in two situations:

- When an application wishes to invoke an operating system service e.g., to write data to a file.
- When a hardware device raises an interrupt to request that a device driver be invoked to handle some state change in the device e.g., the arrival of a network packet.

Both occurrences can be referred to as interrupts, and further classified as either *synchronous* or *asynchronous* respectively depending upon whether their incidence is synchronised with the execution of the application program running when the interrupt occurs. However, for the purposes of further discussion I shall use the term *trap* to denote a synchronous interrupt, and limit the term interrupt to refer only to asynchronous instances. While sharing some similarities there are also important differences between synchronous and asynchronous intrusion.



### 2.3.1 Synchronous Intrusion—Traps

Synchronous intrusion occurs when an application executes a sequence of instructions, culminating in the trap instruction itself, which cause the CPU to switch from the unprivileged execution domain (user level) to the privileged kernel domain. Because the application is in a known state before executing the trap, and is aware that such an event is about to happen, the machine state which must be saved can be carefully controlled by the operating system. This situation is analogous to the set of caller-saved registers in a function calling convention: if the application needs to retain state across the invocation it is responsible for saving it.

This caller-saved model of trap has only become the norm in modern i.e., *RISC*, CPU architectures. Prior architectures, such as early versions of the Intel *x86* CPU, provided trap mechanisms which had exceedingly high overhead due to their complexity, often saving the complete set of CPU registers and loading the saved registers for another task. Although the provision of lower-overhead primitives is important it is certainly not the only factor in determining the operating system-dependent trap overhead. Other tasks which increase the intrusion cost include updating virtual memory control registers, switching to a kernel stack frame, and preparing the environment for entry into the kernel proper.

### 2.3.2 Asynchronous Intrusion—Interrupts

The overhead of an interrupt is much greater than that of a trap for two reasons: first, the asynchronous nature of the interrupt means that the application is in an unknown state when it occurs, so the operating system must be conservative in preserving all state which could be affected by its handling of the interrupt; and second, the signaling used to raise an interrupt usually requires several bus transactions to occur when the interrupt is raised and subsequently acknowledged. These additional penalties can significantly increase the overhead of interrupt handling over trap handling, as shown in Section 2.7 (below).

Both synchronous and asynchronous interrupts can also have an adverse effect upon applications beyond the immediate time penalty. Mogul and Borg [58] showed that context switches due to interrupts significantly reduce the benefits of cache memory, since they interfere with locality of reference.

## 2.4 Correlated vs. Uncorrelated Intrusion

An important consideration when evaluating intrusion is the degree to which it is correlated with the application's behaviour. Correlated intrusion is directly related to the actions performed by an application—synchronous intrusion, for example, is entirely correlated with the application execution since each instance of intrusion occurs when the application invokes an operating system service. Uncorrelated intrusion, on the other hand, occurs regardless of the actions of the application program; timer interrupts are an example of such.

Although there is a spectrum of correlation between application behaviour and aggregate intrusion effects, individual instances of intrusion are usually either perfectly correlated with the application behaviour i.e., intrusion occurs due to application actions, or entirely uncorrelated. As mentioned above, synchronous intrusion is perfectly correlated, while asynchronous intrusion is either perfectly correlated (consider an interrupt due to a network packet received in response to a query sent by the application) or entirely uncorrelated e.g., timer interrupts.

Unbounded uncorrelated intrusion is particularly problematic for applications since it can severely impact the amount of time the application receives for execution (see Section 2.6.1).

## 2.5 Intrusion in Multiprocessor Systems

Multiprocessor systems are particularly prone to mechanism intrusion, since most general-purpose operating systems which are used on such systems are adapted from a uniprocessor base. Thus, in addition to having to coordinate access by multiple independent applications to a single physical resource, the operating system on a multiprocessor system must also enforce this protection across multiple CPUs. The concurrency control required to do so introduces an extra layer of mechanism intrusion on top of that present in a uniprocessor system.

This form of intrusion is particularly problematic if a uniprocessor operating system is adapted naively to support multiple processors. Consider a system which serialises access

to kernel resources by simply enclosing the entire kernel within a single lock structure, as is the case in the Linux 2.0 kernel. If multiple applications spend a significant fraction of their workload executing within the kernel, as might be expected if data transfer is their primary task, then they effectively become serialised by this global lock, and the multiprocessor system performs no better than a uniprocessor. In fact, the additional overhead of the multiprocessor system's concurrency control may actually reduce the performance slightly. This serialisation has been observed in experiments described in Section 5.2.

## 2.6 Intrusion Factor—an Intrusion Metric

As a metric of the degree of intrusion due to an operating system we define the *Intrusion Factor* as follows:

$$IF = \frac{RealExecutionTime}{IdealExecutionTime}$$

where *Real* and *Ideal Execution Time* are, respectively, the execution time of a given operation both with and (perhaps theoretically) without an operating system present.

For example, the UNIX `getpid()` function, to obtain the calling process's process ID, is extremely simple, requiring only the reading of a field from a structure—perhaps a few instructions which are, in practice, negligible compared to the overhead of the system call invocation mechanism. Thus the intrusion factor (*IF*) for *getpid()* on UNIX is very large, almost the worst-case value (the null system call being worse). For more common operations which perform more 'work' within the operating system, rendering the invocation overhead negligible, the IF is likely to be close to unity—obviously an ideal operating system would have the minimal IF of one for all operations.

Thus the IF provides a measure of how much overhead is incurred on a given operation due to the mechanisms of the operating system. One particularly interesting case arises when the IF becomes infinite—however much time is spent executing the operating system no 'work' is ever accomplished on behalf of applications.

### 2.6.1 Livelock—the Curse of Infinite Intrusion Factor

This case of *infinite intrusion factor* is referred to in earlier work by other researchers as *livelock*. The classic example is livelock due to network interrupt processing, as described by Ramakrishnan [74]. There a UNIX kernel was observed to never make any progress in executing application programs because 100% of CPU time was consumed by interrupt handling.

Hence one of the motivations for the Piglet operating system was the elimination of infinite intrusion factor situations, since their presence is a sufficient, albeit not necessary, condition in rendering the host computer vulnerable to denial-of-service attacks. Since infinite IF can only arise due to asynchronous mechanism intrusion i.e., interrupts, alternative mechanisms must be used in the operating system to alleviate this problem.

## 2.7 Evaluating the Cost of Intrusion

In order to provide an indication of the order of magnitude of mechanism intrusion costs simple experiments were performed using the Linux 2.0 operating system kernel running on a dual-200MHz Pentium Pro-based PC. The Linux kernel was instrumented with hooks to record the value of the timestamp counter at pertinent points in the execution path, and a test application executed which measured the number of cycles consumed by both traps and interrupts. Trap overhead was measured by reading the timestamp counter just before and immediately after executing the *sendto()* system call, while interrupt overhead was measured by recording discontinuities in counter values when spinning in a loop while reading that counter.

Table 2.1 shows various time-based metrics calculated from the Linux kernel event log and application timestamp counter readings. The interrupt period, *Period*, represents the total application CPU time consumed by a single interrupt.

While the interrupt period is the simplest measure of OS intrusion it fails to take account of the useful work done by the OS within that period. Judging exactly what is ‘useful’ work is somewhat subjective but we consider it to be all code whose execution directly contributes to the progress of some application. This includes system call, device

Form of intrusion	Time ( $\mu s$ )				Intrusion Factor
	Period	Work	Overhead	Latency	
System call/trap ( <i>sendto()</i> )	32.3	24.8	7.5	4.1	1.30
Network interrupt ( <i>upComplete</i> )	44.3	27.5	15.8	7.6	1.61

Table 2.1: Measured time costs of OS intrusion

Operating System	CPU	Clock/ <i>MHz</i>	System Call Cost	
			Time/ $\mu s$	Cycles
Linux 1.3.37	Pentium Pro	200	3	600
Linux 1.3.38	Alpha 21064A	275	2	550
Unixware 5.4.2	Pentium Pro	200	4	800
DEC OSF/1 3.0	Alpha 21064	150	11	1650

Table 2.2: Comparative costs of system calls

driver, and bottom-half functions (‘soft’ interrupts), but excludes first-level interrupt/trap handlers, context switches, etc. For Table 2.1 the amount of time not incurred as overhead is shown in the *Work* column, with the difference between this figure and the interrupt period i.e., the intrusion cost, being shown as *Overhead*. The *Latency* figure, obtained from the kernel log, shows the time between initiation of the interrupt and the beginning of useful work, while the final column gives the intrusion factor,  $IF = Period/Work$ , as defined earlier.

These numbers were obtained using the Linux operating system running on an *x86* CPU, so one must question whether these results are representative of general-purpose operating systems. In other words, would a different operating system and/or CPU significantly change the intrusion factor? The results presented by McVoy as part of the *lmbench* benchmark suite [49] confirm that this combination of operating system and platform is at least as good, if not better, than similar systems.

Table 2.2 summarises these results: the leftmost 5 columns are reproduced from Table 7 of McVoy’s paper, the rightmost column is simply the system call time (for the simple system call measured by McVoy) in cycles i.e., the product of the preceding two columns.

Linux performs similarly on both *x86* and *Alpha* CPUs, and actually outperforms commercial operating systems, *Unixware* and *OSF/1* respectively, on both platforms. Thus one may conclude that the intrusion factors calculated for Linux on *x86*, at least for system calls, are not tainted by either the choice of operating system or CPU platform. In addition, the predominance of this platform combination in the network appliance market means that the operating system designer is shackled to that platform—peculiarities, inefficiencies and all.

The relatively high IF measured above shows that the cost of intrusion is indeed significant if the frequency of intrusion is high enough. Even for interrupts which cause a large amount of ‘useful work’ to be accomplished, the IF is still fairly high. Thus one of the primary goals for the design of a network appliance operating system is the minimisation of mechanism intrusion cost.

## Chapter 3

# The Piglet Architecture

The preceding chapters described the motivation behind and need for a new operating system architecture specifically designed for network appliances, which can be summarised as:

- Unrealised potential for optimisation due to workload characteristics e.g., emphasis on data movement rather than computation.
- Availability of cheap, high-performance multiprocessor systems which are inefficiently utilised by general-purpose operating systems.
- High overhead of existing operating system architectures, especially for system call invocations which are critical in a network appliance application.

Thus the *Piglet* operating system was designed in response to these motivating factors. Before describing the design of Piglet it is illustrative to first consider some of its predecessors.

### 3.1 The Evolution of Operating Systems

For many years every different computer system had its own operating system. Computers were expensive and slow, so the priorities of an operating system were to maximise the availability of the machine to many users. Examples of such systems include time-sharing

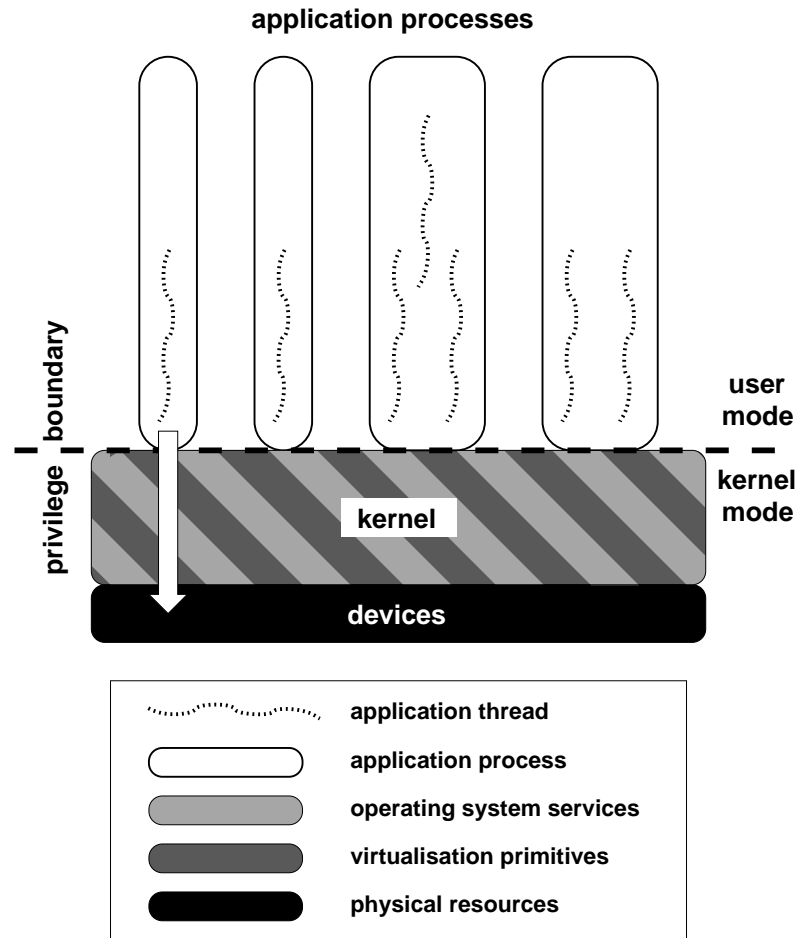


Figure 3.1: Monolithic operating system structure

systems e.g., the *Compatible Time-Sharing System (CTSS)* [15] and *Multics* [68], supporting multiple user applications per machine, and virtual-machine systems e.g., IBM’s *VM/370* [16] and the Bell Labs *MERT* [45], which supported multiple operating systems on a single physical machine. However, in the early 1980s new trends in computer system performance, namely the emergence of the high-performance, RISC CPU-based workstation, drastically changed the computing environment. Now individual users could each have powerful computers for their own use—consequently the requirements of operating systems changed accordingly.

Accompanying this change, and a factor in enabling it, was the increasing dominance



of the UNIX operating system. Because of the ready availability of the Berkeley System Distribution (BSD) source code many workstation vendors developed variants of UNIX as the operating system for their brand of workstation e.g., *HP-UX*, *Solaris*, *Ultrix*, *Irix*. Hence UNIX and its derivatives soon became the predominant operating systems for high-performance workstations and servers, since most workstations vendors also sold servers with the same CPU architecture.

UNIX and all of its derivatives shared an architecture depicted in Figure 3.1, usually referred to as a *monolithic* kernel. The kernel executes in a privileged kernel mode while applications execute in an unprivileged user-mode, thus prohibiting them from direct access to the physical resources. The kernel provides high-level services to applications e.g., network protocol stacks, filesystems, as the only mechanisms by which they may use physical resources. Thus virtualisation is accomplished by abstraction, hiding all of the details of resource management from applications at the same time as sharing resources among them.

In Figure 3.1 each application process is represented by a box enclosing one or more threads of execution. The mapping of threads onto physical CPUs is not shown since the dominant model of *symmetric multiprocessing* treats each CPU equivalently. The path used by an application to access a physical resource is indicated by the large arrow.

### 3.1.1 The Microkernel

A monolithic kernel offers many advantages for general purpose applications, such as a (mostly) uniform programming interface across all UNIX derivatives despite the heterogeneity of the underlying hardware platforms. However, it soon became clear that such a kernel structure was too restrictive for certain applications. Specifically, when the abstractions provided by the kernel did not closely match the requirements of an application then efficiency and performance were usually the casualties. As mentioned in Section 2.2, the best example of this is probably Stonebraker’s 1981 paper titled “*Operating System Support for Database Management*” [85], wherein several examples of such mismatches were encountered when a database management system was deployed on top of a general-purpose operating system.

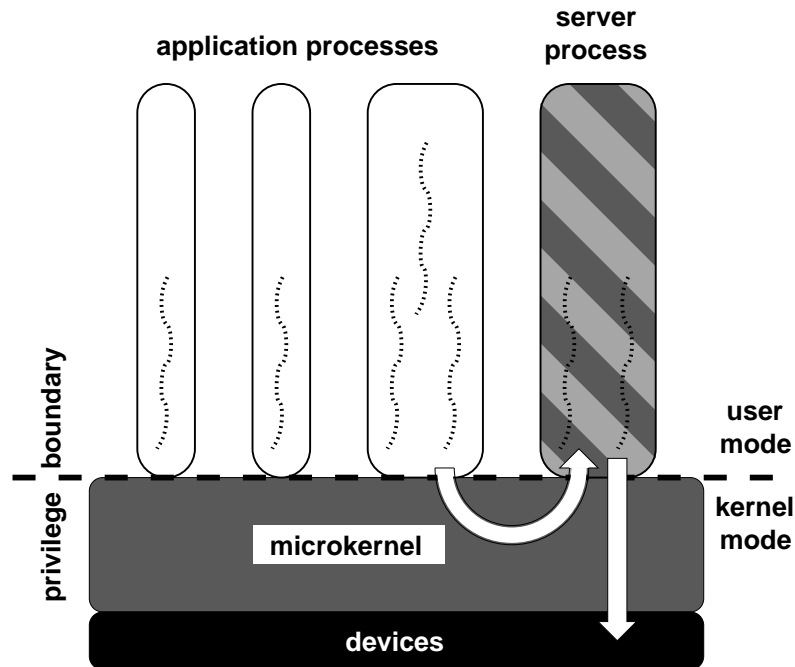


Figure 3.2: Microkernel-based operating system structure

The first attempt to solve this problem was to move operating system services out of the kernel into server processes—the *microkernel* approach, shown in Figure 3.2. In this model the kernel itself implements only the minimum of virtualisation services, delegating the provision of services such as networking and filesystems to server processes. Applications request services from these servers by sending messages to them using kernel-provided *interprocess communication (IPC)* primitives. By virtue of supporting multiple servers, each providing a different set of services, the operating system can offer different resource management strategies to different applications, thus avoiding the “*one-size-fits-all*” problem.

While the microkernel approach promised flexibility in resource management, early microkernel implementations were crippled by the high overheads of IPC. Although later microkernel systems, such as *L3* [43] and *L4* [29], vastly reduced the overhead of IPC, other groups, such as the MIT *Exokernel* project [21], Cambridge’s *Nemesis* kernel [42, 5],

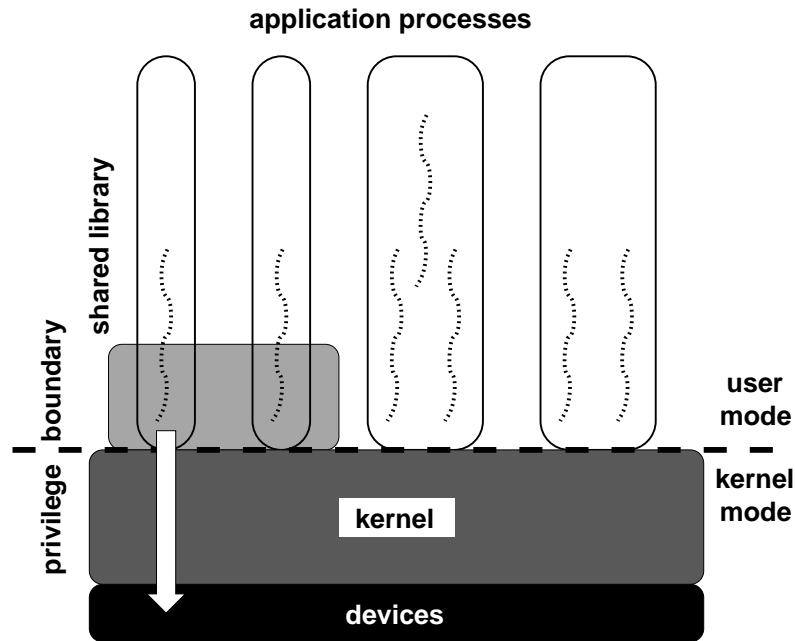


Figure 3.3: Vertically-structured operating system

and Stanford’s *Cache Kernel* [11], instead proposed an alternative solution—the *vertically-structured* operating system.

### 3.1.2 Vertically-Structured Systems

Proponents of vertically-structured systems pointed out two flaws in the microkernel design: the overhead of IPC, and the scheduling problems caused by executing all service code in application-level servers on behalf of other applications. More specifically, when an application sends a request to a server it is necessary to transfer the application’s scheduling properties e.g., priority, to the server, and resource usage by the server be must accounted correctly to the application. While some mechanisms have been proposed to deal with this problem, such as Mercer’s *processor reserves* [52], other researchers instead proposed abandoning the microkernel architecture in favour of their vertically-structured systems.

In a vertically-structured system, as shown in Figure 3.3, the kernel still provides only the lowest-level virtualisation primitives. In contrast to a microkernel, however, the kernel

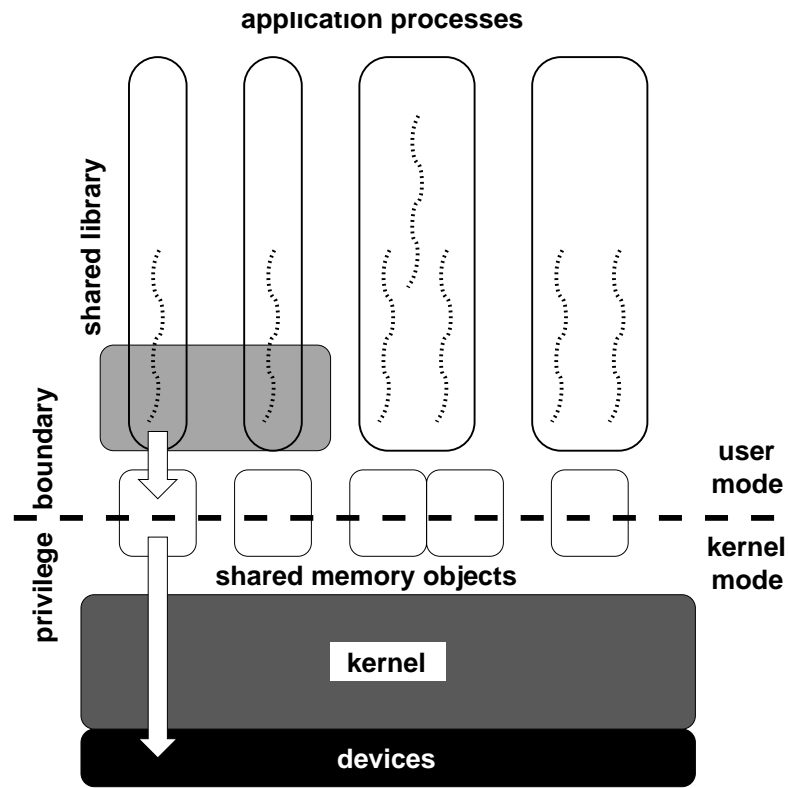


Figure 3.4: Active kernel-based operating system

also provides applications with restricted but direct i.e., not indirectly through a server, access to physical resources using low-level functions. In this way applications still retain the ability to implement their own higher-level resource management functions, but without having to act through an intermediary. Thus IPC overhead is eliminated and resources can be accounted for more easily. Of course, general-purpose implementations of many system services suffice for the majority of applications, so shared libraries are often an integral part of a vertically-structured system in order to provide those common functions.

### 3.1.3 The Next Step—an Active Kernel

Vertically-structured systems offer the cleanest separation between virtualisation primitives and higher-level service functions, thus minimising policy intrusion, but they fail to

address one of the problems motivating the need for a network appliance-specific operating system—the overheads of mechanism intrusion. Consideration of the fundamental cause of those overheads leads to the basis for an operating system structure which addresses that problem.

In every prior operating system architecture—monolithic, microkernel or vertically-structured—the kernel itself is a passive object, essentially a shared library with special semantic properties e.g., executing in a privileged execution domain. As described earlier (Section 2.3) the mechanisms used to enforce these semantics, such as privilege boundaries, place an associated cost upon applications. The observation that these overheads are inherent in a system where applications must frequently cross the privilege boundary lead to an obvious conclusion: only by obviating the requirement to cross the privilege boundary can the overheads be eliminated.

One system structure that has been proposed which removes this requirement is to have applications execute in the same domain i.e., at the same privilege level, as the operating system itself. Such systems fall into two classes: those where protection between applications is enforced by the language in which the applications are written e.g., *Java*-based operating systems, such as the University of Utah’s *KaffeOS* [4]; and systems with no protection between applications e.g., *MS-DOS*. Unfortunately, network appliances fit into neither class—their applications are not written in the appropriate languages for the first class, and isolating application programs is obviously important for reasons of safety, security and reliability. Hence the total elimination of privilege boundaries is not a viable option.

Instead, a new operating system structure is proposed: the *Active Kernel*. Rather than eliminating privilege boundaries, the active kernel instead permits communication between applications and itself without either party crossing the privilege boundary. This is accomplished by using communication channels which can bridge the privilege boundary—shared memory objects. Applications post messages to these objects requesting that a particular function be performed, the kernel subsequently retrieves the messages and processes the request. Since these shared objects are entirely passive the action of an application posting to one cannot be used to initiate the execution of the kernel without making the

objects protected, thus re-introducing the privilege-crossing overhead. The alternative solution adopted is to have the kernel execute continuously, polling shared objects for posted messages—hence the *active* moniker. The first architecture for such an operating system is named *Piglet*—hereafter we use Piglet as a reference example of an active kernel architecture.

## 3.2 Anatomy of Piglet

The structure of Piglet is shown in Figure 3.4. Much of the architecture is similar to a vertically-structured system, specifically the separation of virtualisation primitives, provided by the kernel, and higher-level functions, implemented directly by applications and shared-libraries. Piglet is significantly different, however, in the mechanism by which low-level operations are invoked by applications: instead of executing a trap to cross the privilege boundary and execute kernel code directly a message is posted to a shared-memory object; this message is subsequently retrieved and processed by the kernel.

An obvious practical implication of the Piglet architecture is that a processor must be dedicated to making the kernel active, so the architecture is only applicable to multiprocessor systems. As described in the introduction to this thesis though, such systems are becoming more commonplace and hence this does not seem too restrictive a requirement. In addition, since Piglet is targeted specifically at network appliances, rather than general-purpose applications, the architecture can exploit specific characteristics of that target application domain, for example the high frequency of system call invocation. This is especially relevant to the multiprocessor requirement in light of experimental evidence (Section 5.2) showing that kernel-intensive workloads do not scale as well on a multiprocessor under a conventional operating system as their computationally-intensive brethren.

### 3.2.1 Primary Features of Piglet

Piglet’s key features, those arising from the motivating factors summarised at the beginning of this chapter, are:

1. *Dedicated kernel processor*—kernel runs continuously.

2. *Interrupt-free kernel*—kernel is never interrupted.
3. *Shared-memory communication*—low-overhead service invocation.

Taken together these three features are responsible for the most significant benefits derived from the architecture. A dedicated kernel processor does not directly provide performance or other benefits, but facilitates the other key design strategies. Eliminating interrupts from the kernel drastically reduces complexity and also eliminates a significant source of mechanism intrusion. The use of shared-memory communication between applications and the kernel is crucial in providing a minimal-overhead service invocation mechanism, but also alters the programming model.

### 3.2.2 Secondary Architectural Features

Before describing these key features of the Piglet architecture in more detail it is helpful to summarise other aspects of the kernel design so as to place the central characteristics within a complete system context.

- *Commodity hardware platform*: Piglet is designed to be hardware-neutral, since being tied-in to a specific CPU platform or devices exposes the system to obsolescence if the hardware does not evolve as fast as commodity technologies.
- *Language independence*: although programming languages such as *Java*, *TAL* [24, 60] and *Cyclone* [33, 32] can offer many advantages to systems programmers e.g., type safety, verifiability, run-time code generation, many network appliance applications do not or cannot take advantage of them, often due to a large established codebase or performance factors. Thus Piglet is designed to be language-independent.
- *Capability protection model*: Piglet uses a capability-based protection model, inspired by the *EROS* [80] kernel, to provide access control, protection and security. In light of security vulnerabilities discovered in common appliance applications, for example the multiple flaws recently unearthed in the BIND application [9], a more robust security model than that provided by UNIX may help minimise the impact of such problems.

At this point I will proceed to describe the three key principles of the Piglet architecture, before detailing the prototype implementation of Piglet and illustrating some operational properties of the system with an in-depth examination of the network subsystem.

### 3.3 A Dedicated Kernel Processor

The first and most fundamental design decision made in Piglet is that the kernel runs continuously on a dedicated processor. This is necessitated by the elimination of interrupts and the use of the shared-memory communication model, and differentiates Piglet from other operating system architectures.

A kernel which runs continuously without interrupts can be made extremely simple—at its most basic level it consists of a single thread of execution which performs kernel functions *ad infinitum*. Simplicity is derived from not having to design the kernel to handle interrupts occurring at inopportune moments, and not having to explicitly make kernel state accessible to trap and interrupt handlers. This principle will be discussed in detail later (Section 3.4), but is important to bear in mind when describing the design of the kernel.

The set of functions which must be performed by this top-level function are essentially the same as those executed by a conventional operating system kernel. However, the nature of the Piglet communication mechanism, also described later (Section 3.5), means that Piglet must poll for service requests rather than waiting until an application initiates processing by issuing a trap instruction. Thus the structure of the top-level function, which shall be referred to hereafter as *main* by analogy with the top-level function in a C program, consists of the composition of a number of polling modules, each of which is responsible for a different section of the kernel.

Figure 3.5 shows a simple representation of the first-level modules which comprise the Piglet kernel. It is important to point out that other modules could, and would, be incorporated into *main* to handle other subsystems e.g., block devices, but only a representative subset are shown here. Several of these modules are described in detail subsequently, when the architecture of the Piglet network subsystem is introduced, but let



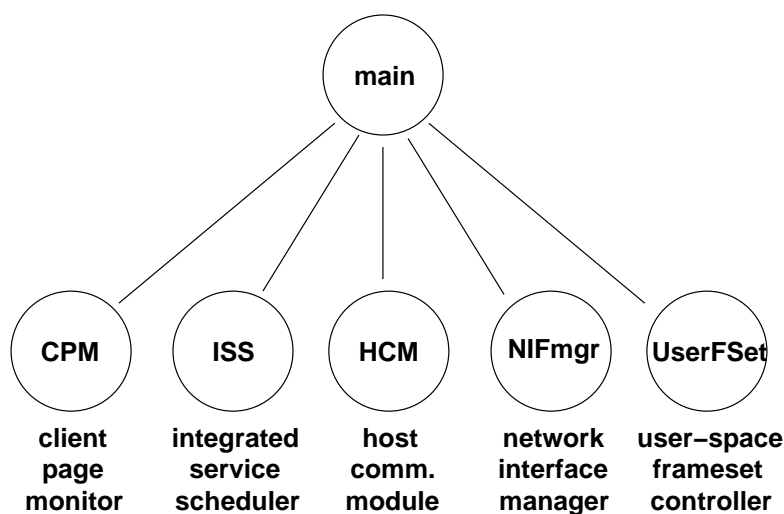


Figure 3.5: Structure of the *main* function

us examine the others now.

### 3.3.1 Client Page Monitor

The *Client Page Monitor* provides communication between the Piglet kernel and its clients e.g., threads of application programs. The client page is a shared-memory object which is mapped into both kernel and client address spaces, thus permitting either to read or write from the the client page at any time. The client requests operations be performed by writing *Posted Service Requests* to the client page. The client page monitor subsequently polls the client page and retrieves those PSRs. Once the operation requested by the PSR has completed the kernel returns status information via the client page. The operation of the client page monitor is expanded upon in the description of shared-memory communication in Section 3.5.

### 3.3.2 Integrated Service Scheduler

The *Integrated Service Scheduler* is responsible for executing additional functions other than the first-level polling modules which are integrated into the Piglet kernel. These functions are encapsulated into *service modules*, an example group of which are shown

Service	Periodic Function	Rx Handler	Tx SBlk
ARP module	Yes	Yes	Yes
ICMP module	No	Yes	No
Virtual Clock packet scheduler	Yes	No	Yes

Table 3.1: Network-related services in the Piglet kernel

in Table 3.1. For each service the table shows which of three functions it provides: a periodically-executed *service thread*, a packet reception (Rx) handler, and a packet transmit (Tx) *service block (SBlk)*. The periodic function is executed by the scheduler either after an elapsed time, or when the state of the service changes. The packet reception handler is installed by the service into the Piglet packet filter to demultiplex service-specific packets appropriately; the transmit SBlk can be used by applications to have the service perform special processing on packets to be transmitted.

As examples of the type of processing performed by these service functions consider the services shown in Table 3.1. For the purposes of this description it is sufficient to state that a frameset is the system object used by both the application and kernel to represent a network flow endpoint; see Section 4.2 for a more detailed description. The ARP module fulfills dual roles: it handles ARP requests sent to the Piglet system, and prepends a link-level header to packets before transmission. Incoming requests are processed by the Rx handler, link-level headers prepended by the ARP SBlk. The latter is obviously a common requirement, but the flexibility to have this function applied only to framesets with the SBlk attached is useful for network protocols which supply the link-level header themselves e.g., ARP itself. Finally, the periodic function is used to update ARP cache entries and flush stale ones.

The other services shown in Table 3.1 are fairly self-explanatory: the ICMP service is used to process ICMP packets received by the Piglet kernel from the network, while the Virtual Clock service provides an SBlk which can be used to apply the *Virtual Clock* packet scheduling algorithm [92] to outgoing packets; other SBlks could be used to provide alternative packet-scheduling mechanisms e.g., token bucket, leaky bucket. The ability to

perform packet scheduling is obviously important in providing QoS guarantees, as demonstrated in an early implementation of Piglet [63].

One final noteworthy feature of the service scheduler is the use of separate stack frames for each service to provide a rudimentary form of cooperative multithreading. When scheduling a service's periodic function the scheduler switches from the main Piglet stack frame to the service's private stack frame—the application yields control back to the scheduler when either processing is completed or a 'reasonable' amount of time has passed. The question of how long a service should execute before yielding to the scheduler is obviously complex, but must be short enough that Piglet remains responsive to external requests. Such a form of cooperative scheduling is thus only feasible for integrated services which can be trusted to behave as good citizens, and not appropriate for arbitrary code modules. The usefulness of private stack frames, which permit an application to yield control at almost any point in execution without having to explicitly save state, was demonstrated in the Piglet MPEG video decoder [64]—when the frame being decoded is incomplete the service yields to the scheduler until the Rx handler notifies the scheduler that new data has arrived.

### 3.3.3 Host Communication Module

The final module shown in Figure 3.5, the *Host Communication Module*, is required when Piglet is implemented in a hybrid system alongside a conventional operating system. Detailed description of this design is deferred until Chapter 4, but in essence both operating systems coexist on a shared hardware platform. Because both execute in kernel-mode i.e., with direct access to physical resources, it is important to coordinate the use of those resources. The host communications module accomplishes this task in Piglet—see Section 4.1.3 for a description.

### 3.3.4 Scheduling in the Polling Function

An important consideration in the design of *main* is the weight which is given to each first-level polling function. In the simplest design each would be executed equally frequently, but this is obviously not the only strategy. A relatively simple modification is to attach

static weights to each function, perhaps based upon expected system workloads—an infrastructure appliance, for example, spends a greater proportion of CPU time processing network packets than control application requests, so the device polling function should be assigned a greater weight than the client page monitor. A more sophisticated enhancement would be to introduce dynamic adaptation into the polling function—as the system workload evolves over time *main* can adjust the function weights accordingly. Unfortunately such enhancements are beyond the scope of this thesis.

The structure of the polling function is crucial to the behaviour of the system. A compromise must be struck between polling objects sufficiently frequently to guarantee acceptable response time to changes of state, but not wasting too much time polling objects which infrequently require processing. Thus one can view the composition of *main* as a classic scheduling problem where each component of *main* is a scheduled entity with its own properties e.g., deadline, priority, period.

Extending the polling function beyond the basic round-robin approach, whether by the simple enhancement of attaching a weight to each function, or implementing a fully-featured scheduler, introduces additional complexity into a system which had previously possessed simplicity as a key virtue. However, it is important to recognise that this complexity takes a different form from that which is so troublesome in an interrupt-driven kernel, the complexity arising from having to deal with interrupts in a multitude of locations. Scheduling complexity is a design challenge—the implementation of a scheduler for polling functions is likely to be relatively simple, so the likelihood of introducing obscure bugs is low. Interrupt-handling complexity is an implementation challenge though—if the designer/implementor makes a mistake then the resulting bugs often render the system unstable.

### 3.4 No Asynchronous Interrupts

The second primary design decision incorporated into Piglet is the elimination of asynchronous interrupts. The motivation to do so was twofold: firstly, the overhead of the

interrupt mechanism is typically very high—recall the intrusion factor of 1.61 from Section 2.7; and secondly, asynchronous interrupts also add significantly to the complexity of an operating system kernel. This complexity arises from the need to update internal data structures atomically in order to guarantee that the kernel’s internal state remains consistent. The most common approach to doing so is to disable interrupts while executing these *critical regions*, however this approach is often error-prone, since the programmer must be aware of every possible interaction between interrupts handlers and internal data structures, and can increase interrupt latency.

Compelling evidence of the complexity and error-prone nature of interrupt-driven kernels is given by Engler’s paper [20] discussing the use of ‘*meta-level compilation*’ to detect semantic errors in the Linux 2.3.99 kernel. Of particular relevance is the use by the compiler of simple, high-level specifications to detect violations of semantic constraints e.g., inconsistent interrupt states. This technique uncovered approximately 60 errors where a function which could block was called with interrupts disabled, potentially leading to deadlock, and another 40 situations where the interrupt state was not correctly restored when abruptly returning from a function.

Thus it was decided at an early stage in the Piglet design that asynchronous interrupts were to be avoided whenever reasonably possible. Interrupts are only to be used to indicate catastrophic errors e.g., a hardware failure, where the kernel is not expected to be able to recover gracefully. Since the majority of input-output devices normally use interrupts to request processing from the kernel i.e., the execution of a device driver, an alternative mechanism must be used in Piglet. The active nature of the Piglet kernel makes polling the obvious choice.

This change effectively inverts the conventional relationship between a device and its driver: rather than the device requesting that the driver perform some operation on its behalf only when the device state changes the driver instead frequently queries the state of the device to determine which, if any, functions to execute. An important concern in such a scheme is maintaining the kernel’s responsiveness to device state changes—how frequently must a driver poll the associated device? Many existing devices were designed under the assumption that interrupts are the primary means of communication with the

operating system (exceptions include the *UPenn* ATM adapter [88]), and consequently that prompt processing can be triggered by raising an interrupt. Hence the nature of the device often dictates a relatively high polling frequency. On the other hand, the increasing sophistication of devices has reduced this problem: as an example, the adoption of direct memory access (DMA) by devices in preference to programmed I/O (PIO) has reduced the interrupt burden placed on the kernel since the driver need not be invoked to transfer every unit of data from to/from the device's buffers.

One possible hurdle in the elimination of interrupts is the possibility that some devices may only be able to signal certain state changes by raising an interrupt, thus making polling not entirely effective as a substitute. It is anticipated that such devices would be handled by creating a service module that would be marked as needing to be scheduled by a simple 'stub' interrupt handler which has no access to general kernel data structures. Such an approach is taken in the *Nemesis* kernel [42], where interrupts are handled by *schedulable regions*, and *Swift* [12], where interrupt handlers schedule a process with a real-time deadline to perform the actual work.

### 3.5 Shared-Memory Communication

The final, but perhaps most important, design decision embodied by Piglet was that shared memory objects would be used as the primary communication mechanism between application programs and the kernel. The motivation for this was the observation that the conventional mechanism of traps imposes overhead on the application due to the privilege boundary which must be crossed by the trap. Shared memory eliminates this overhead since the object used for communication is mapped directly into the application's address space.

Using shared-memory objects as the primary communication channel in a conventional passive kernel architecture is not possible since an active entity to service those objects will not generally exist. The active kernel facilitates this mode of communication since the shared objects can be polled sufficiently often to provide the application with reasonable guarantees of service promptness.

As described above, the primary task of the Piglet kernel is polling applications and devices to service their processing requests. In order to maintain overall responsiveness the kernel must be able to poll objects with very low overhead, else polling itself introduces costs similar to those which the architecture is designed to reduce, namely the extra cost to determine when a specific function should be performed on top of the cost of performing that function.

The design of the various shared-memory objects to be used for communication is of the utmost importance. Several properties must be achieved:

- *Simplicity*: a complex communication protocol would negate many of the advantages obtained by eliminating the trap overhead.
- *Low common-case cost*: the common case when an object is polled is to perform no action, so this determination must be quick and easy.
- *Lock-free*: the application and Piglet access objects concurrently, creating obvious potential for conflict. Locking to enforce consistency should be avoided due to the negative implications of having an application lock a kernel data structure.
- *Compact*: one solution to the problem of concurrent access to objects is to copy them to a private location, so making the objects small is advantageous.

In addition to these factors, the context in which the shared object structures are to be employed must also be taken into consideration. For example, network packets passed from kernel to application or vice versa could potentially form an arbitrarily long sequence, so a structure which is not inherently of fixed length is preferable.

The data structure which was deemed to be most suitable was a queue using non-blocking synchronisation methods [53] to provide internal consistency. Ring buffers also meet all of the primary criteria but their fixed size precludes them from being used as network queues, and it was deemed undesirable to have a multiplicity of shared object structures. The design of the queue is such that it can be accessed by multiple writers without additional concurrency control, but only a single reader is permissible—the case

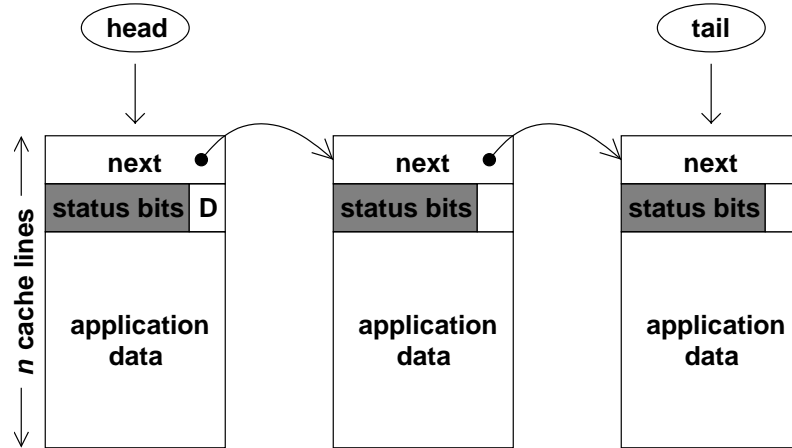


Figure 3.6: Basic structure of the Piglet shared queue object

of multiple readers is not considered common enough to necessitate being handled directly by the queue.

The basic queue structure is shown in Figure 3.6. Associated with each queue are head and tail pointers—the head points to the first/oldest element in the queue, the tail points to the last/most recently added element. At least one element is always retained in the queue, so both pointers are always valid. Each element is an integer number of cache lines in size: violating this condition creates false sharing between queue elements when the kernel and application access adjacent elements concurrently, imposing unnecessary cache coherency misses. The first word of each element is a pointer to the next element; the second word is used for element status, further explained below; the remaining words are available for application-specific use, typically as a message payload when each element constitutes a message.

The operation of the queue is as follows:

1. The queue is initialised with an empty element. The contents are usually unimportant as long as the *next* pointer is null (indicating no subsequent element) and the *done* (*D*) bit is set in the status word. Both head and tail pointers point to this element.
2. A new element is enqueued by first preparing the element in memory, atomically



exchanging the tail pointer with the address of this element, then updating the *next* pointer of the old tail element to point to the new element.

3. The queue is polled by first checking the *done* bit: if it is not set then the head element is returned. If the bit is set, indicating the element has already been processed, the *next* pointer in the head element is checked: if it is null then no new element is available, otherwise the head pointer is updated and the new element returned to the polling function.
4. Once an element has been processed the processing function sets the *done* bit if the element should not be processed again—an example of a situation when an element is not marked done is when many queues are polled but only one of them is selected for processing.

Consistency of the queue between multiple writers is maintained by using an atomic exchange operation e.g., the *xchg* instruction in the *x86* architecture or a load-linked/store-conditional pair on most RISC architectures, to update the tail pointer: consider two writers *A* and *B* simultaneously enqueueing elements *a* and *b* respectively. Since the exchange operation is atomic, and the CPUs serialise access to a shared memory location, one of the two writers must execute its exchange operation first. Assume it is *A*, which then updates the old tail element's *next* pointer to point to *a*. Immediately after *A*'s exchange completes *B*'s will proceed—the new tail pointer, *a*, is exchanged for *b* and *a*'s *next* pointer updated to *b*. This dictates that applications must set the *next* pointer to null before performing the exchange and not change the pointer subsequently. There is also a window of time within which the queue becomes temporarily detached i.e., between exchange completing and the old tail element's *next* being updated. This is not problematic from the reader's perspective since the old tail's *next* pointer is null before being updated, so the queue appears to have not been updated. However, if an interrupt occurs in this window of time then consistency may not be restored until after a large delay.

This queue design places two constraints upon both readers and writers: queue elements need to be accessible to all parties, either at the same address in all address spaces

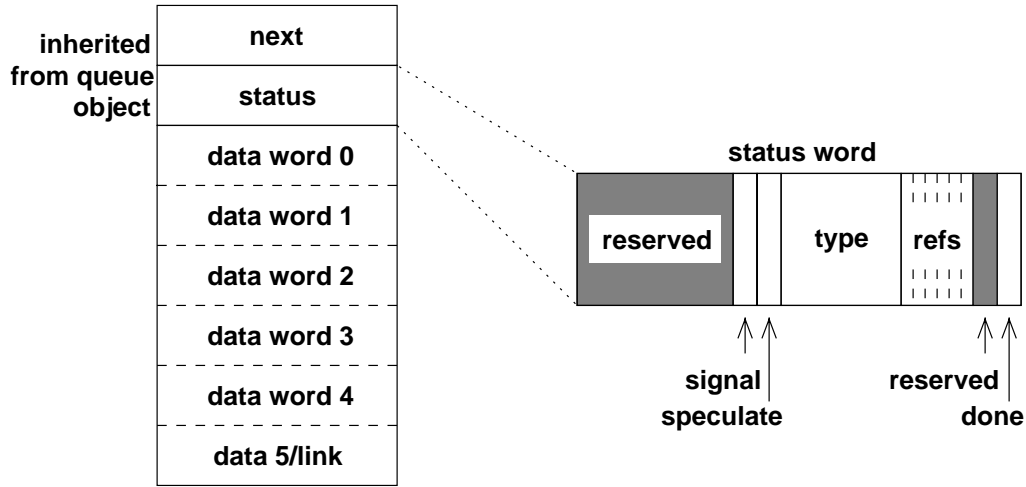


Figure 3.7: Structure of the Piglet Posted Service Request

or by translation; and both parties must adhere to the above protocol. Obviously a malicious entity could easily corrupt the queue, but the simplicity of the structure makes such corruption easy to detect.

This structure has been used in several places in the current kernel implementation, most significantly as the mechanism by which applications invoke kernel services, and as the fundamental building block for a network communication endpoint. These examples are described in the following section and Section 4.2 respectively.

### 3.6 The Piglet-Application Interface

A key characteristic of any operating system is the programming interface it exposes to application programs. In conventional operating systems the synchronous trap mechanism used closely resembles a function call interface, so system calls can most naturally be presented to applications in that manner. An active kernel, however, presents an altogether different programming model due to the fundamentally different mechanism used for communication. Note that the issue here is one of low-level semantics of the interface rather than the details of what functions it provides; unfortunately the boundary between these two properties is often blurred.

### 3.6.1 API Basics

In Piglet the application programming interface is constructed on top of the shared memory queues described above. As stated earlier, every application thread has a client page associated with it, either private to that thread or shared with other threads. The client page consists primarily of a shared queue into which the application can post messages—*Posted Service Requests (PSRs)*—requesting that the kernel performs a specified operation on the application's behalf. The structure of a PSR is shown in Figure 3.7. The client page contains a fixed-size array of PSRs in order to eliminate the need for allocation and deallocation.

Each PSR occupies exactly one cache line, 32 bytes on the *x86* CPUs used in the prototype implementation. The first two words of the structure are inherited from the shared queue structure, although only the *done* bit is reserved by the shared queue protocol. The remaining six words in the PSR are used to pass data between application and kernel, with the last word also being used in some circumstances as a link pointer. The functions of the various unreserved bits in the status word are as follows:

- *Reference bits (6)* are used to indicate whether the content of the corresponding data word is a value or reference. References are used by an application when constructing sequences of PSRs, their use is described further later.
- *Type field* is used to specify the operation being requested by the application.
- *Speculate bit* specifies whether the processing of this PSR is conditional upon the success of a preceding PSR operation. If so then the *link* pointer (word 5) points to the preceding PSR.
- *Signal bit* is set when the kernel should send a signal to the application upon completion of this operation.

The simplest examples of a PSR are created by setting the type field to indicate the operation being invoked and passing the values of the operation's parameters in the data words. Piglet uses capability references to identify kernel objects e.g., network communication endpoints, virtual memory regions, physical memory pages; a capability reference

can be represented in a single data word. After processing the PSR the kernel stores a return code and other operation-specific data in the data words for subsequent retrieval by the application.

### 3.6.2 Asynchronicity Creates Concurrency

While the use of PSRs appears similar to a function-call interface, the biggest difference arises from the nature of the shared queue object to which PSRs are posted. Because the queue is merely a container for elements the act of posting does not immediately initiate kernel processing. Instead, the PSR is processed when the kernel next polls the client page, thus introducing a decoupling between application and kernel. This asynchronicity of service invocation is the biggest difference between an active kernel API and that of a conventional operating system, raising interesting questions about application programming models.

An asynchronous programming model has both advantages and disadvantages when compared to a conventional synchronous interface. The primary advantage is that the decoupling provides implicit concurrency between the execution of the application and the kernel. However, the application programmer is faced with an unfamiliar programming model and must restructure their application to obtain maximum benefit from this concurrency.

A practical side-effect of having an asynchronous service invocation mechanism is that notification of the success or failure of an operation is no longer a simple matter of returning the appropriate code since the application is unlikely to be waiting for that return value. While Piglet does update each PSR with a completion code which the application can poll to determine when the operation has completed and its outcome, the kernel also uses asynchronous signals, analogous to UNIX signals, to indicate operation completion. Normally a signal is only sent to the application if an operation fails, but the *signal* bit can be set to request a signal even if the operation is successful. This facility can be used by an application to block until the PSR completes if it cannot proceed without the result.

### 3.6.3 References Arguments and Speculation

This programming model would perform very poorly if an application had to post one PSR and wait for the results it generates before being able to use them in the next PSR. Hence Piglet provides two features to reduce such inefficiency: reference arguments and speculative processing.

The *reference bits* in the status word allow an application to designate particular arguments as being references rather than values. A reference argument specifies a data word of another PSR as the target of the reference, which is dereferenced by the kernel when it processes the referring PSR. Reference arguments permit that PSR to use values which were unknown to the application at the time it posted the PSR to the client page. An example of this usage occurs when one PSR allocates a kernel resource which is then used by a subsequent PSR—the capability reference for the kernel resource is created during processing of the first PSR so the value is not known to the application at the time that it posts the second.

*Speculative processing* is used by an application to notify the kernel that one PSR is dependent upon another—if the first was not processed successfully then the second should be ignored. Speculation is useful in two distinct situations: when the second PSR is contingent upon either a state change or output produced by the first, and when the application does not wish to perform the second operation if the first did not succeed, even though it could. This latter case arises when two operations refer to a sequence of data—if the first unit was not transmitted on the network (say) then there is no point transmitting the second.

The use of both references and speculation are revisited in Section 4.2, which describes the Piglet networking subsystem in detail and includes an example sequence of dependent PSRs.

### 3.6.4 Optimistic Service Invocation

The ability to chain posted service requests together in this manner is only beneficial to the application if the likelihood of an invocation failing, and thus negating execution of all subsequent elements in the chain, is low. This is a general question of interface

design rather than a specific Piglet issue, but one important point gleaned from the Piglet programming interface is worth mentioning.

First, consider the most common causes of system call failure (other than invalid input): insufficient resources and access prohibition. Both failure modes usually arise from a single class of function, namely resource allocation. Therefore, in order to segregate the most common causes of failure, high-level functions should be decomposed to isolate resource allocation from resource manipulation. This is in fact a natural distinction since the two types of operation are usually not intermingled but rather separated into disjoint phases of a higher-level operation. Thus an application can construct a sequence of resource manipulation primitives with reasonable confidence that failure is unlikely.

This programming model fits conveniently into the paradigm espoused by proponents of vertically-structured systems, of which Piglet is an example. Such systems only implement low-level primitives within the kernel, thus imposing a decomposition of high-level functions. A good example of the separation of resource allocation from manipulation is *Nemesis's Application Device Channel* architecture [5]: resource allocation is performed by an application-level server which establishes appropriate channels for the client application to directly manipulate resources.

## Chapter 4

# Implementing Piglet

A prototype implementation of Piglet provides the best means of evaluating the design principles described in the previous section. Constructing an entire operating system from the ground up requires a large amount of ‘grunt’ work, consequently the Piglet prototype was engineered as an extension to a host operating system, thus creating a hybrid architecture.

Another practical decision taken early on was to only implement a single functional subsystem in the Piglet prototype, one that is sufficient to evaluate the key characteristics of Piglet. Construction of a hybrid system permits delegation of the remaining core functions e.g., filesystem, processes, virtual memory, etc., to the host system in order to support the test framework. Since Piglet was designed from the very beginning as an operating system for network appliances the obvious choice of functionality to implement in the prototype is the network subsystem.

### 4.1 Building the Linux/Piglet Hybrid

Linux was selected as the host operating system for the first Piglet implementation, primarily because it had the most mature support for multiprocessor systems at the time the prototype was begun. It is important to note though that Piglet could coexist equally well with any other host operating system, such as one of the ‘free’ 4.4BSD variants.

The Piglet prototype was built to coexist with the Linux kernel, allowing applications to

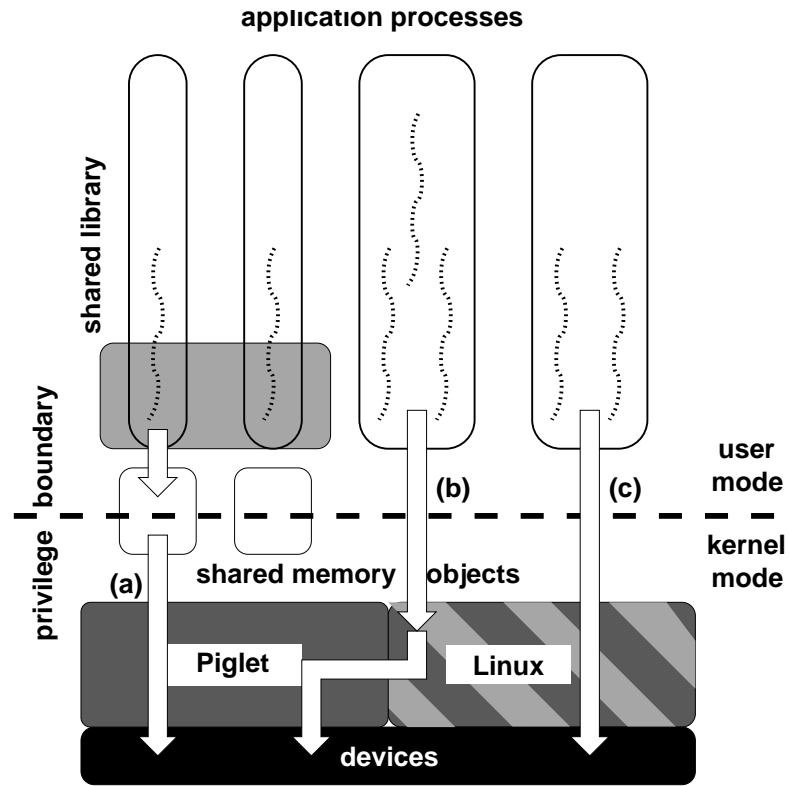


Figure 4.1: Hybrid Linux/Piglet kernel structure

utilise functionality from both kernels. The system was designed such that Piglet would be hidden from the Linux kernel, except for a number of well-placed hooks into the Linux kernel which are used by Piglet to acquire physical resources e.g., memory, CPUs. Apart from the connections to those hooks the Piglet prototype is essentially a faithful implementation of the architecture described earlier. Figure 4.1 shows the structure of the Linux/Piglet hybrid. The arrows labeled *a*, *b*, and *c* respectively represent applications accessing devices in three different ways: directly through Piglet, through Linux then Piglet, and directly through Linux.



### 4.1.1 Linux Kernel Hooks

Supporting Piglet alongside the Linux host required minor changes be made to the kernel itself; the small scale of these required changes can be credited in part to the reasonably modular structure of the Linux kernel. Modifications were made in only three areas:

- *Scheduling*: Piglet requires a CPU dedicated to the active kernel, so the scheduler must be modified to support such an allocation.
- *Concurrency control*: a small number of data structures must potentially be accessed concurrently by both Piglet and Linux, necessitating additional concurrency control.
- *External interfaces*: the interface exported by the Linux kernel is intended for use by device drivers, not another operating system, so must be enhanced with the additional necessary functions.

The scheduler was modified in two ways: the addition of a data structure storing the set of schedulable and interruptible CPUs, and the ability to force a particular task (kernel thread) to be executed on a given processor. These two features are used to initialise the Piglet kernel by removing a processor from the schedulable set and forcing it to run the Piglet kernel thread.

Additional concurrency control beyond that used by Linux was required due to the simple nature of the mechanisms already put in place. In the version of the kernel that Piglet is based upon, 2.0.30, SMP support is somewhat primitive so a single, global lock is used to serialise access to the kernel. In the hybrid prototype it is necessary to run Piglet ‘outside’ this kernel lock, but certain data structures must also be accessed by Piglet, thus violating the kernel’s assumption that, at any given time, only one CPU is executing kernel code i.e., accessing those structures. An example of such a structure is the run-queue—when Piglet needs to make an application thread runnable it must update the run-queue, but the kernel may be doing so simultaneously. In order to keep the prototype implementation simple a spin-lock is used to serialise access to the run-queue.

Minor interface changes were necessary because Piglet requires more services from the host operating system than a typical device driver e.g., the ability to send inter-processor

interrupts, and functions to release and reacquire the global kernel lock. This usually entailed nothing further than adding those functions to the kernel's exported symbol tables.

The other features Piglet required from the Linux kernel, mostly needed in the initialisation process, were readily implemented using existing functions and data structures—Piglet appears to Linux as an ordinary block device. For example, Piglet is dynamically loaded as a module when a control program accesses the appropriate block device file, the module initialisation function is used to acquire memory and a CPU for the Piglet thread, and finally device `ioctl` functions are used to configure and run the Piglet kernel.

#### 4.1.2 Using Host Services Within Piglet

The biggest advantage obtained by constructing the Piglet kernel to coexist alongside Linux is that many of the Linux kernel's functions can be used by Piglet. At the simplest level, Linux kernel functions are utilised to perform routine or simple functions in Piglet, such as page-table manipulation and the formatting of log messages. More important, however, is Piglet's ability to share application processes with Linux, thus permitting those applications to use services provided by both kernels. This ability was essential in writing programs to evaluate the Piglet services without having to also develop every other aspect of the system to a level sufficient to (say) load and execute programs entirely using Piglet functions.

One particularly challenging problem arising from the sharing of processes is how to schedule those processes in a manner which maintains consistency with the Linux scheduler. For example, consider a process which is receiving network packets from Piglet. The process checks to see if a packet is available—if not, that process blocks until a packet is received. Unfortunately a race condition exists between checking the network queue status and the application being blocked—consider a case where Piglet receives a packet just after the application checks its queue, sees that the application is still runnable and so updates the queue status but doesn't send a signal, thus causing the application to block unnecessarily, perhaps indefinitely. However, such a fix would require considerable modification of the Linux scheduling mechanism, so instead Piglet is implemented to act conservatively and always send the wakeup signal; this is essentially the same behaviour as a conventional

network interface that raises an interrupt regardless of what the kernel may be doing.

Piglet also uses *fast wakeup* signals to optimise scheduling in certain cases. Applications can install a fast wakeup interrupt handler which can be used by Piglet to reschedule the application without the overhead of the usual Linux mechanisms e.g., acquiring the kernel lock, running the scheduler, restoring unnecessary state. When a signal is to be sent to an application e.g., due to an error processing a PSR, or network packet reception, if Piglet can determine that the application blocked itself using the `piglet_block` system call then it sends the appropriate inter-processor interrupt vector to one of the application CPUs, causing the fast wakeup handler to be executed which switches context directly to the now-runnable application i.e., without invoking the scheduler. Since the fast wakeup handler does not acquire the kernel lock it must determine that no other CPU is currently executing in the kernel—if any CPU is holding the kernel lock the fast wakeup handler instead calls the regular *reschedule* interrupt handler.

Memory allocation is also tricky in a dual-kernel environment due to the need to maintain consistent data about how physical memory pages are being used. For the prototype implementation Piglet acquires large chunks of memory from the Linux kernel at initialisation, then uses its own internal memory allocator to subdivide these chunks into page-sized or smaller blocks. Access to memory allocated by Piglet is occasionally required from Linux system calls e.g., when creating client pages, thus requiring that the memory allocation functions be synchronised. To minimise the number of synchronised accesses performed by the Piglet *main* function a service module maintains a cache of private memory blocks for use within the Piglet kernel.

### 4.1.3 The Host Communications Module

As mentioned in Chapter 3, the hybrid implementation of Piglet communicates with the Linux kernel using a service module, the *Host Communications Module (HCM)*. This module performs two main functions: maintaining TLB consistency with Linux and providing a virtual device interface through which Linux may access those network devices managed by Piglet.

Because the *x86* architecture does not provide hardware mechanisms for maintaining

TLB consistency it is the responsibility of the operating system to do so. In the symmetric multiprocessing Linux kernel this is accomplished using broadcast inter-processor interrupts, sent by the CPU updating the page tables, which cause all other CPUs to flush their TLBs. Because the Piglet kernel is designed to be interrupt-free, this mechanism cannot be used to keep Piglet consistent with the Linux kernel. Fortunately, a solution already existed in Linux: because a spinlock is used to synchronise access to the kernel, including to execute interrupt handlers, the system would deadlock if one CPU modified a page table entry, sent “invalidate TLB” messages to the other CPUs, then waited for them to acknowledge this action, since none of them would be able to enter the kernel to acknowledge the interrupt. Linux solves this problem by having CPUs check a global memory flag while attempting to acquire the kernel spinlock—if the flag is set then the TLB is invalidated and the request acknowledged before continuing to attempt to acquire the lock. Piglet just uses the same mechanism, polling the status word every time the HCM service function is executed.

The second major function of the HCM is to provide Linux with a virtual device through which it can access the Piglet network devices. This is necessary because it is not possible to have two device drivers, Piglet and Linux, both attempting to control a single physical device. Instead, the Piglet driver is used and a generic Ethernet device presented to the Linux kernel. This can be used for both sending packets from non-Piglet applications through the Piglet network devices, and also when Piglet receives network packets that it does not know how to handle—the HCM provides a default receive handler which passes packets to the Linux virtual interface.

## 4.2 The Piglet Network Subsystem

At this point let us consider the structure of the prototype network subsystem as a detailed example of how architectural features of Piglet are implemented and used. Figure 4.2 shows this structure, representing both data flow and module interactions. Transmitted packets flow from the client to the network interface controller (NIC) down the left-hand side of the figure, received packets flow up the right-hand side.

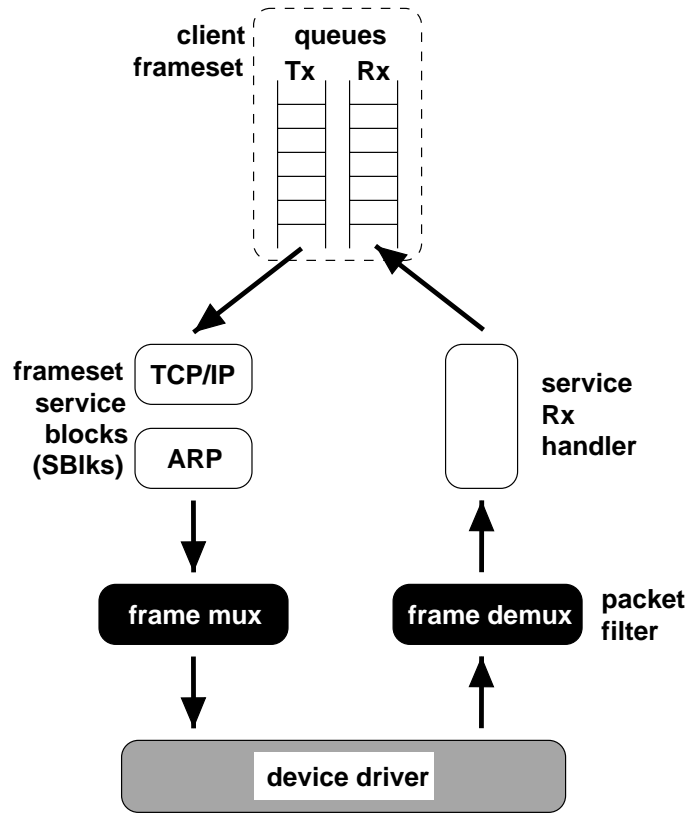


Figure 4.2: Network module structure

Piglet clients, either application programs or a host operating system e.g., Linux in the hybrid implementation, use network services through a *frameset*—the fundamental network communication endpoint in Piglet. Clients obtain framesets from the kernel using the posted service request mechanism (see below, Section 4.2.3), subsequently the frameset itself is used as a shared memory object for direct communication between the client and Piglet. Each frameset consists of internal control fields (not shown), plus two shared memory queues that hold transmitted and received *frames*. Each frame, comprising a *frame header* and a small number of possibly non-contiguous data buffers, represents a single network packet; Piglet does not perform fragmentation of frames into packets. The structure of both framesets and frame headers is depicted in Figure 4.3.

Only the client-visible part of the frameset structure is shown in Figure 4.3. The queue

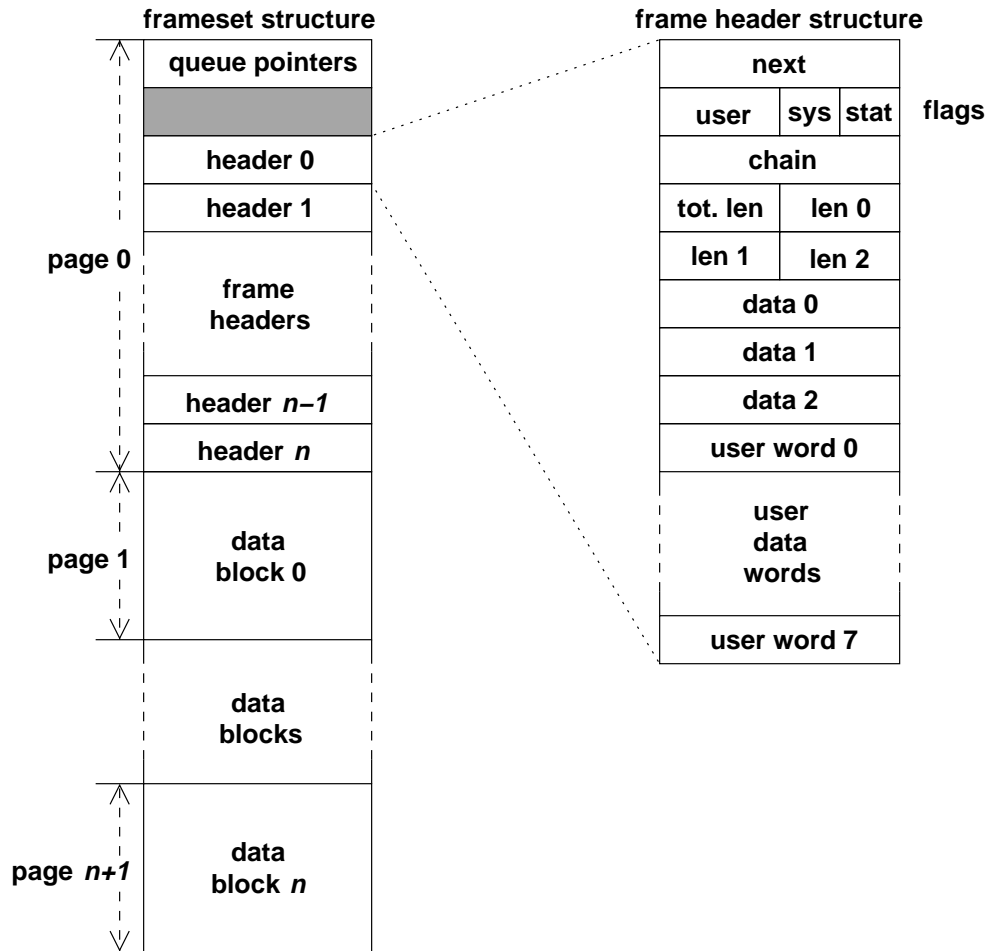


Figure 4.3: Frameset and user-level frame header structure

pointers—tail pointer for the transmit queue, head for the receive queue—are stored at the beginning of the structure; the complementary pointers i.e., transmit head and receive tail, are required only by the Piglet kernel, so are not made accessible to the client. Following some miscellaneous frameset state fields, are  $n$  frame headers each with the structure shown on the right side of the figure. Finally, a virtual mapping for  $n$  pages is placed after the headers, one page per header, although not all virtual pages are backed by physical pages at any given time.

Part of this structure is dictated by the prototype implementation using very simple

mappings from headers to corresponding data blocks for user-level framesets. Such a mapping is necessary due to the fact that a physical page of memory has a different virtual address from each of the application and kernel perspectives—a legacy of the hybrid Linux environment. Piglet thus constrains each frame header to data contained in the corresponding virtual page (as seen from the application perspective), and uses the frame header index to find the data block. This constraint could easily be relaxed in future implementations.

Each frame header consists of a number of fields. Figure 4.3 shows these fields for a user-level frame header: for a kernel frame header the first half of the header is the same, but the user words are replaced with auxiliary fields e.g., reference count, timestamp. Only the common fields are of particular interest, so the internal kernel fields are not discussed further.

The *next* pointer is inherited from the common queue element structure described earlier, but frame headers use a status byte rather than a single *done* bit; the function of this status byte is described below. System flags are reserved for kernel use, while the user flags field is available for application-specific purposes. The chain pointer is used to create linked lists of frame headers—this is a common enough occurrence in protocol stack implementations e.g., a TCP retransmit queue, IP refragmentation queue, etc., to merit a dedicated field. Four half-word sized length fields indicate the total length of the packet and the length of each of up to three data blocks, pointers to which are stored in the final three words. The ability to specify three non-contiguous data buffers is used to perform *scatter-gather DMA* when transferring data to/from network devices.

As a frame is transmitted or received it undergoes various state changes. This state must be exposed to both the Piglet kernel and the associated application so that it can manage its use of the headers. Figure 4.4 shows the state transition diagram associated with a frame. Frames can be in one of 8 states, represented as circles, encoded as shown within the circle. All valid transitions are shown on the graph: the text label associated with each transition indicates the action which triggers the transition, and a C-syntax expression indicating how the state variable is atomically modified by that action; *cmpxchg(x, pending, skipped)* means “if the current value of *x* is *pending* then change it to *skipped*, otherwise

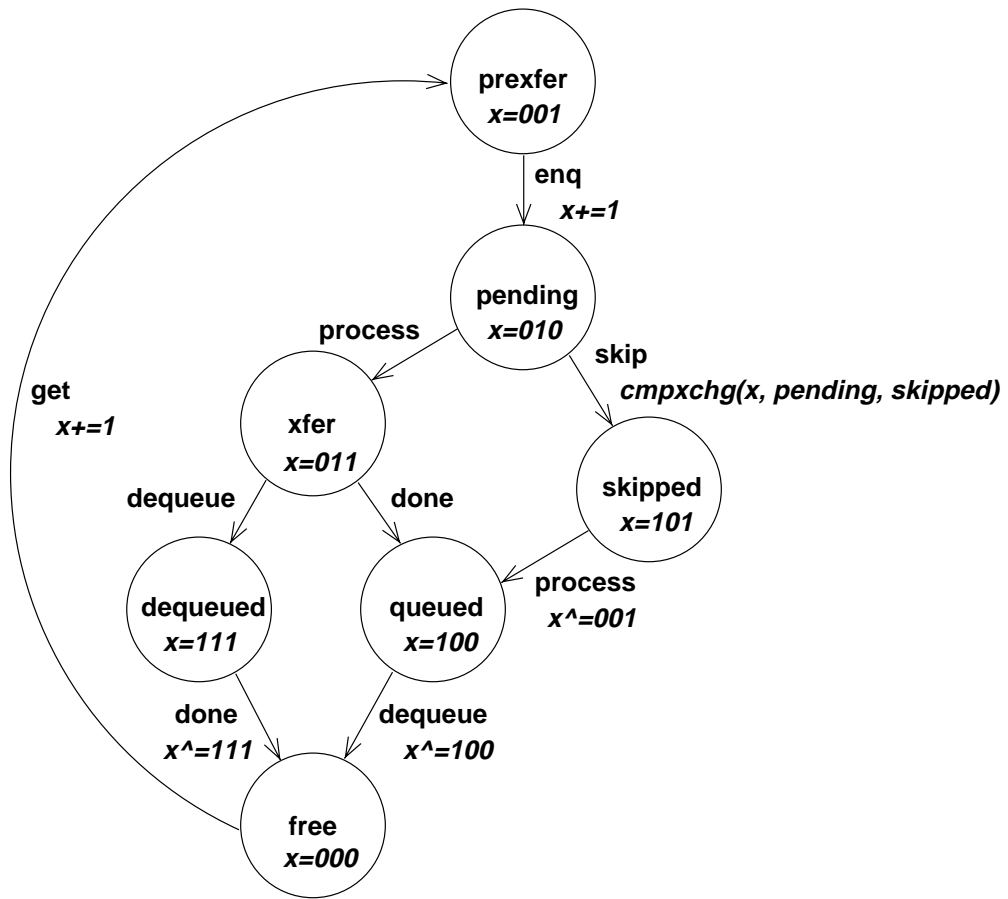


Figure 4.4: Frame header state transitions

do nothing”, executed atomically. The meaning of each action is as follows:

- *get*: mark a free frame as being in-use by the application.
- *enq*: enqueue a frame which has been prepared by the application.
- *process*: begin processing the frame e.g., pass to the network device for transmission.
- *cancel*: skip processing if not already begun.
- *done*: frame processing complete.
- *dequeue*: remove frame header from queue.



Most of the states themselves are self-explanatory, however the steps by which a frame moves from *pending* to *free* are worth explaining. Consider a frame being transmitted by an application: after being enqueued on the transmit queue and subsequently selected by the Piglet kernel for transmission the *process* action is used to indicate that the frame is being processed e.g., is being transferred by DMA to the network device. In this state the frame header structure is being used by both the network device, to conduct the DMA transfer, and the queue reader and writer(s), as the head element.

One of two possible events can occur next: if processing completes before another frame is enqueued then *done* is used to indicate that the network device no longer needs access to the frame—however the application cannot reuse this frame yet since it still constitutes the head of the queue. Alternatively, another frame could be enqueued, permitting the *dequeue* operation to be performed—in this state though the device is still accessing the frame, so again it cannot be reused yet. Only when both *done* and *dequeue* have been performed is the frame available for reuse; by careful choice of the state representations and the transition operations every action can be executed without having to first determine what state the element is currently in, thus eliminating a potential race condition.

#### 4.2.1 Frame Transmission

Frames are transmitted by enqueueing them onto the Tx queue, which is later polled by the kernel *frame multiplexer*. When the multiplexer determines that a new frame has been enqueued it creates a shadow frame header by copying the user-level frame header into kernel memory inaccessible to the user-space application. This step is required to prevent modification of the frame header fields once the frame is passed to the device, it is not required when the client is trusted e.g., a host operating system or internal client such as the ICMP service.

After creating the shadow frame header and attaching the data buffers the frame is passed to the service blocks associated with the frameset. These can perform various manipulations upon the frame but at a minimum create the appropriate headers for transmission of the frame e.g., TCP, IP, and link-level. To prevent modification of packet headers by the application program these headers are created in kernel memory, optionally using

parameters passed in the frame headers, for example the TCP sequence number. Piglet uses gather DMA when transferring the packet to the NIC to avoid having to copy the packet headers and payload into a contiguous physical memory buffer.

Service blocks also perform the task of packet scheduling. Such blocks attach a transmit timestamp to each frame which is then used by the frame multiplexer to select one frame per NIC from the set of framesets associated with that NIC. This frame is finally passed to the NIC's device driver which creates the NIC-specific descriptors used to construct the packet, then initiates DMA of the packet from application memory to the device.

#### 4.2.2 Frame Reception

A network packet received by a device is processed in essentially the same manner. Piglet's *main* function frequently calls the device driver's *poll* function to process received frames; if the driver determines that DMA of a received packet has completed then it passes the packet to the packet filter for demultiplexing.

Many existing implementations of packet filters exist, from the original, simple *Berkeley Packet Filter* [48] to the advanced *Dynamic Packet Filter* [22], all of which could be incorporated into Piglet, but for practical reasons the Piglet prototype instead uses a very simple scheme. The filter first checks the link-level header to determine the packet type—ARP is handled by the ARP service handler, IP is passed to the next step, all other types are processed using the default handler as described later. For IP packets the destination IP address is first checked: if it is not the correct address for this interface the packet is passed to a forwarding service, otherwise various fields are extracted from the IP and higher-level protocol headers, then their values used to lookup the packet handler function in a tree structure. Values are used in the following order, most significant first:

1. IP protocol
2. Destination port (if applicable)
3. Source port (if applicable)
4. Source IP address

The IP protocol serves to differentiate packets by higher-level protocol, and for the common transport protocols—TCP and UDP—the destination port is the next best differentiator. For TCP connections initiated by this system, when the source port number is usually unspecified and thus randomly generated, this field alone normally suffices for unambiguous demultiplexing, although obviously the other fields must be checked for equality. For connections initiated by the remote host the destination port is usually a well-known value e.g., port 80 for HTTP servers, so packets from many sources will be addressed to a single destination port. In such cases the source port is a better differentiator than source address, since even for different remote hosts the normal random selection of source ports means that it is unlikely two remote senders will use the same source port number. If this should happen then the source IP address can be used to separate packets to the correct frameset.

If a packet matches in the filter then a tuple of service Rx handler and frameset is returned—the latter piece of data is supplied as an argument, along with the packet header, to the indicated Rx handler. If a packet did not match in the filter, or the Rx handler returns a special value to indicate that it cannot process the packet, a default action is performed, usually to drop the packet but in the hybrid environment the host communications module registers a default handler which passes the packet to the host operating system.

The action performed by the Rx handler function is service-specific but usually results in a frame being enqueued on a frameset. For user-level framesets this entails finding a header which is not already being used, using the state protocol described above, then initialising the header and modifying the application's corresponding virtual memory page to map the physical memory page containing the packet. Finally a signal is sent to the client if necessary, the form of which depends upon the client e.g., a virtual device interrupt for the Linux kernel, a reschedule or *fast wakeup* interrupt for applications.

It is worth acknowledging at this point that page remapping in this manner is not entirely satisfactory: in order to prevent one application from having access to another's data through page mappings it is necessary to ensure that only one packet gets received into each physical page, thus causing inefficient use of receive buffers, particularly for

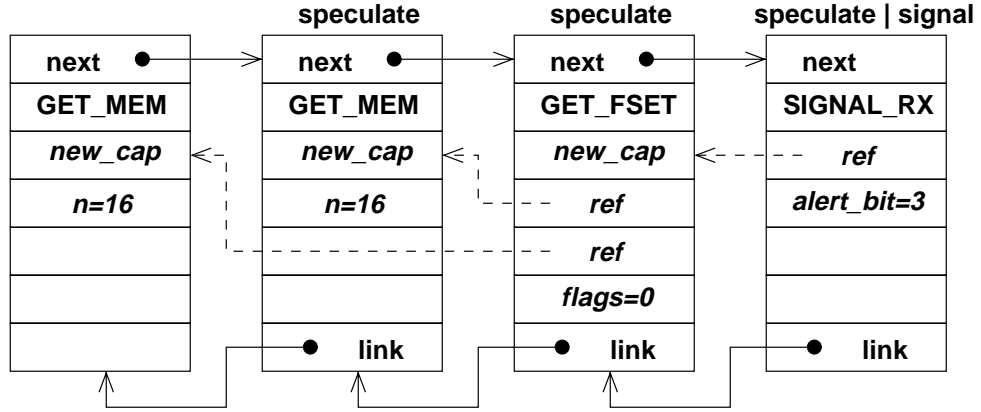


Figure 4.5: Frameset creation: an example of chained posted service requests

small packets. There are several solutions to this problem: one possibility, investigated by Geoff Milord [54], uses the *x86* segmentation hardware to provide arbitrary-granularity protection boundaries on received buffers; another option is to use network devices which are capable of directly demultiplexing packets to the correct application e.g. Pratt’s *Arsenic* device [73].

### 4.2.3 Frameset Creation

The final aspect of the network subsystem worth considering in detail is the means by which applications create framesets. While not particularly interesting from a networking perspective it provides a good example of how *speculation* and *reference arguments* are used by an application to combine multiple posted service requests into a single logical unit. Figure 4.5 shows the chain of PSRs used in this example.

An application wishing to create a frameset first must acquire memory capabilities for both the Tx and Rx buffer pages. Each of these can be acquired from the kernel using a *GET\_MEM* posted service request, specifying the number of pages to allocate to the memory capability. Since this is a new capability being created the capability reference passed is *new\_cap*, a placeholder for the returned capability. If the first allocation fails there is no point attempting the second, so the second request is marked as being speculative

and linked to the first.

The application then posts the *GET\_FSET* PSR, which requires the capability references for each of the buffers be passed as arguments, along with an integer value indicating creation flags e.g., whether the frameset should be shared or private. To permit the application to post this request without waiting for the prior requests to complete, the memory capability references are passed as reference arguments pointing to the new capability to be returned by the earlier request. This request is also marked speculative and linked to the second memory allocation.

Finally, an application which anticipates blocking on packet reception can request that Piglet send a signal when a new packet is enqueued on the frameset's receive queue. This request requires a frameset capability reference, which must also be passed as a reference argument since the frameset is not yet allocated, and an integer indicating which alert bit (analogous to a UNIX signal number) should be marked when the signal is sent. This request is contingent upon frameset creation so it is marked speculative and linked, and, because it is the last in a group, the signal bit is set, notifying Piglet to send the *system call complete* signal when this PSR has been processed.

Chaining PSRs together in this way permits an application to issue a logically related series of requests in one group, without waiting for the result of each in turn. Such a feature is critical in minimising the amount of time the application spends waiting for PSRs to be processed.

## Chapter 5

# Evaluating Piglet

The prototype implementation of Piglet serves two purposes: it provides a vehicle for testing the feasibility of design features, and acts as a means of evaluating the performance of a Piglet system on typical operating system functions. Much of the previous chapter demonstrated the usefulness of the prototype in system design, this chapter and the following one describe various experiments used to characterise the behaviour of the Piglet prototype, referred to subsequently as Piglet.

There are obviously an extremely large number of performance characterisations which could be performed, but this thesis only addresses a number of key metrics:

- *The overhead of service invocation* how much overhead is introduced by the kernel's mechanisms for system service invocation.
- *Throughput and concurrency of data transfer* how well the kernel supports efficient transfer of data from multiple independent applications to shared physical devices.
- *Characterisation of the polling function* measuring the response and processing times of the *main* polling function.
- *Evaluating application-level performance* how the system as a whole performs as an application platform.

This set of performance measurements can be divided into two groups: *microbenchmarks* and *macrobenchmarks*. Microbenchmarks, such as McVoy's *lmbench* suite [49],

evaluate aspects of the target system in isolation, while macrobenchmarks measure system performance on a coarse-grained scale, typically by running ‘real’ applications e.g., the *SPEC* group’s *CPU2000* [31] set of benchmarks. The first three experiments in the list above fall into the microbenchmark category and are described in this chapter; application-level performance is discussed in Chapter 6.

## 5.1 The Overhead of Service Invocation

A fundamental property of a network appliance is the predominance of data transfer operations over computation, as stated in Section 1.2. In any operating system which one might consider as a basis for a network appliance, whether monolithic, vertically-structured, or some other architecture such as Piglet, applications must ultimately invoke system services to transfer data to and from I/O devices. These system services may be high-level system calls in a UNIX-like system, or lower-level primitives in a vertically-structured system, but a system service invocation must be performed in either case. Thus the performance of the service invocation mechanism can have a significant impact upon application performance—as shown earlier (see Section 2.7) the *intrusion factor* for a network transmit system call in Linux is 1.30 i.e., an overhead of 30% on such operations.

Two different comparisons were made between the Piglet network subsystem and the equivalent Linux kernel protocol stack. A coarse-grained measurement of operation latency was obtained using the standard `ping` application to measure round-trip latency. A more detailed analysis of the overheads of packet transmit and receive was performed by instrumenting both protocol stacks with profiling hooks.

### 5.1.1 Measurement of Round-Trip Latency

In order to compare the overhead of the Piglet service invocation mechanism—*Posted Service Requests*—with that of a conventional system a simple network latency measurement was used. A Piglet protocol stack equivalent to UNIX ‘raw’ sockets was created as a user-space library and linked with the standard `ping` application. `ping` was chosen because of its simplicity, allowing the impact of system-calls to be easily isolated. Several

```
ping muffin2 -f -c 100000 -s 64 --use-tsc 200 --histogram >/dev/null
```

where: `muffin2`            name of target machine  
`-f`                        flood-ping mode  
`-c 100000`                send 100000 packets  
`-s 64`                    add payload of 64 bytes  
`--use-tsc 200`          use timestamp counter for timing, 200 cycles per  $\mu s$   
`--histogram`            generate a histogram of timing frequencies  
`>/dev/null`              redirect output to `/dev/null`

Figure 5.1: Command used to measure round-trip latencies.

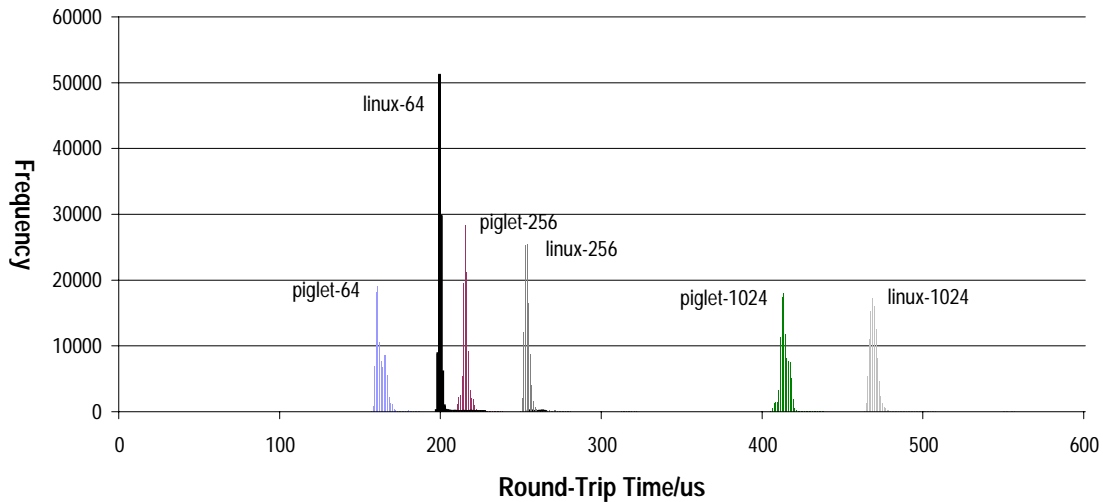


Figure 5.2: Distribution of round-trip times as a function of payload size

modifications were made to the program:

- Support for the Piglet user-space network interface was added. This entailed replacing socket calls with the equivalent calls to the Piglet user-space library.
- The CPU's timestamp counter was used for measuring round-trip time (RTT) since this can be read directly from user-space, while *gettimeofday()* requires a system-call invocation.
- A profiling option was added to log the value of the timestamp counter at various points in execution.
- A option to generate a histogram of RTT distribution was added.



Payload/bytes	Modal RTT/ $\mu s$	
	Linux	Piglet
64	199	161
256	253	215
1024	468	413

Table 5.1: Mean round-trip times

`ping` was invoked as shown in Figure 5.1, with the addition of the `--piglet` parameter when running the Piglet tests. Three different payload sizes were used, and the experiment was repeated several times to ensure that results were representative. The histograms generated are shown in Figure 5.2, with the label *piglet-n* attached to the histogram for Piglet with  $n$  payload bytes, and similarly for *linux-n*.

The histograms show the observed frequency against RTT for the given payload size. The position and shape of the histograms are more significant than the frequency values themselves. Modal RTTs i.e., the position of the most-frequently observed RTT (highest column in the histogram) are shown in Table 5.1 as an aid to comparison, with the leftmost histograms having smallest RTTs. Taller, narrower histograms indicate smaller variance in the distribution of round-trip times.

The most significant point to take from Figure 5.2 is that the round-trip times for the Piglet system are lower than those for the Linux system by approximately  $38\mu s$  for 64-byte payloads, and up to  $55\mu s$  for 1024-byte payloads. This dependence upon payload size arises because the Piglet user-space protocol stack performs DMA of the packet payload (not headers) directly from application memory while Linux copies into a kernel buffer from which DMA is performed; the cost of copying is negligible for 64-byte payloads however so  $38\mu$  can be taken as a representative figure.

The only difference between the Piglet and Linux tests is which protocol stack is used; in both cases the application runs as a Linux process, using the default scheduling and other kernel features. Thus it seems reasonable to infer that the difference in measured round-trip time is entirely due to differences in how network services are provided by the kernel. In order to further analyse this  $38\mu s$  difference it is necessary to use a more

sophisticated experiment.

### 5.1.2 Analysis of System-Call Overheads

In order to perform a more detailed analysis event logs were generated by profiling hooks embedded in the Piglet network subsystem and added to the Linux kernel protocol stack. These hooks record the value of the CPU's timestamp counter upon entry to and exit from every function used in the transmission and reception of network packets. Timestamp values are stored in memory so as not to impact application behaviour with the overhead of writing to an external log, and can be retrieved using special functions later. Representative sections of the profile for a single round-trip were extracted and are displayed graphically in Figures 5.3 and 5.4 respectively for Linux and Piglet.

The graphical representation shows the relative value of the cycle counter corresponding to various events in the sending and reception of a single packet. The counter value when the top-level function used to send the ICMP echo packet is called is used as the zero point; the *got\_packet* event indicates when the reply packet was received by the application.

The vertical line down the centre of each figure represent the time axis, with time progressing downwards. The unshaded boxes to the right of the axis represent functions executed in the application, while the shaded boxes to the left represent kernel functions. Stacked boxes represent nested function calls. Each dotted line indicates that the event with the given label occurred at the specified time. For the Linux trace certain labels have been omitted which are not relevant to the analysis; the complete Linux trace is given in Appendix A.

The Piglet trace represents the execution of the kernel processor on the leftmost line. Since the Piglet trace is more concise every event is represented in Figure 5.4.

What is most obvious from these traces is the difference in the time taken by the application to send a packet i.e., from *begin\_send* to *end\_send*. While Linux takes  $\approx 6300$  cycles, Piglet only requires 230. This major difference is due to the application in Piglet only having to post the frame to the transmit queue rather than executing the system call itself.

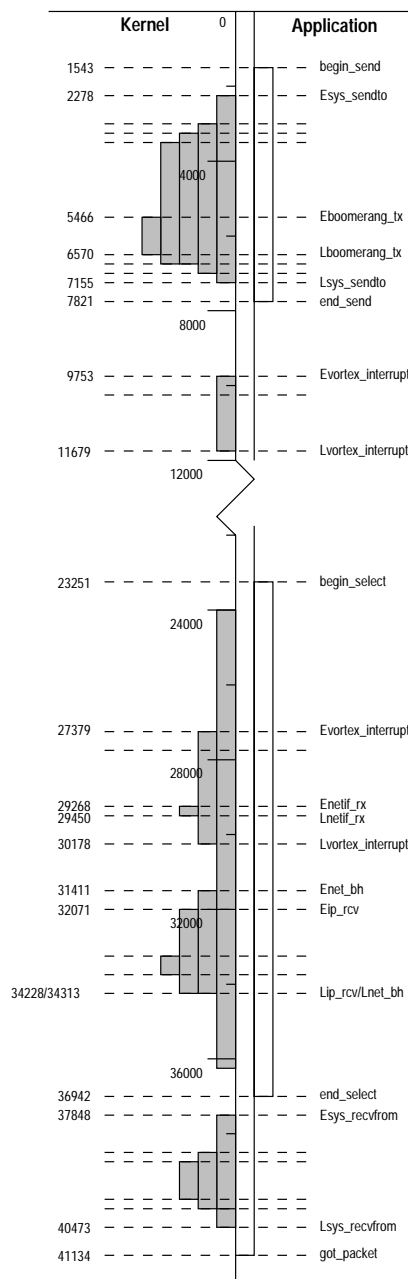


Figure 5.3: Linux ping trace

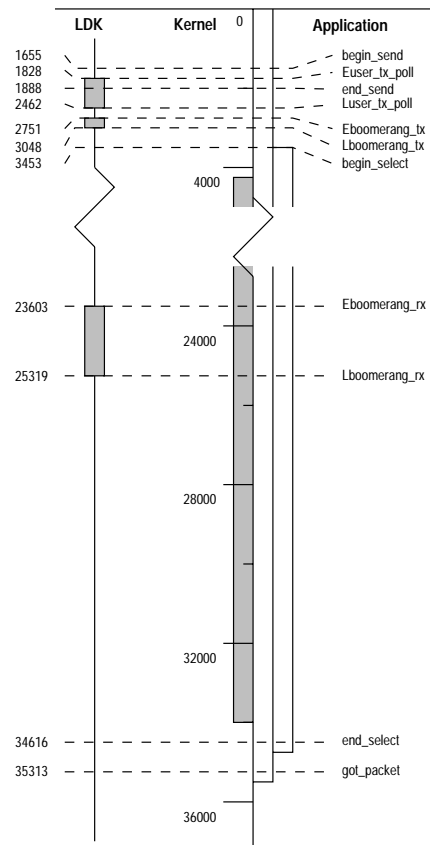


Figure 5.4: Piglet ping trace

The latency between *begin\_send* and the packet having been passed to the network interface (*Lboomerang\_tx*) is also much lower in Piglet—1400 cycles as opposed to 5000. The extra 3600 cycles incurred in the Linux kernel corresponds to an additional contribution of  $18\mu s$  (with a 200MHz CPU clock) to the round-trip time.

As seen from the traces, the biggest factor in this difference is the structure of Linux's protocol stack. Because Linux implements a reasonably high-level interface (BSD sockets) it executes multiple levels of functions (*sys\_sendto*—the generic socket layer; *inet\_sendmsg*—the Internet domain socket family; *raw\_sendto*—the Internet domain raw socket; *ip\_build\_xmit*—build IP headers for the packet) before finally passing the packet to the network interface. The substantial overhead of these multiple layers of abstraction contributes the largest amount to the system call cost in Linux, and provides a significant motivation, and supporting evidence, for vertically-structured operating systems.

While not affecting the round-trip time, the traces also provide an example of mechanism intrusion due to device interrupts. In Linux the first *vortex\_interrupt* event is caused by the NIC raising an interrupt to inform the OS that the packet has been sent; in Piglet the application does not suffer this intrusion because the kernel polls the NIC to determine when packet transmission is complete.

After sending the packet, the **ping** application calls *select()* to block until the reply is received. Since the time when the kernel enters the corresponding top-level function is not recorded, it has been estimated using the measurements from Section 2.7. Notification that the packet has been received and transferred into memory by the network interface occurs in the second *vortex\_interrupt* event for Linux, and the *boomerang\_rx* event for Piglet. The time between that event and *select()* returning to the application is greater in Piglet because the kernel must send a reschedule interrupt to the Linux kernel to indicate that a packet was received.

The *select()* call could be removed in both the Linux and Piglet tests. For Linux it is unnecessary since the *recvfrom()* function blocks if no packet is available; if the call to *select()* is removed then the Linux RTT decreases by  $\approx 20\mu s$ . In the Piglet environment it should be replaced with operations that use the Piglet exception and scheduling mechanisms, doing so leads to a similar reduction in RTT.

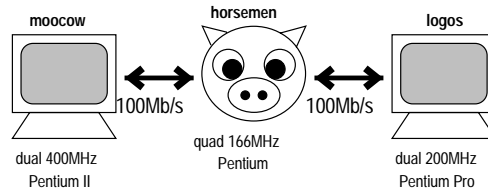


Figure 5.5: Testbed configuration used for bandwidth measurement

Finally, upon returning from *select()* the application receives the packet; in Linux this entails making a *recvfrom()* system call, while in Piglet the application removes a frame header from the frameset’s receive queue. The time difference here (*end\_select* to *got\_packet*,  $21.0\mu s$  vs.  $3.5\mu s$ —a difference of  $17.5\mu s$ ) again demonstrates the costs of system call overhead and multiple levels of abstraction.

This analysis demonstrates two key points. Firstly, the overhead of a system call, 1500 cycles, is significant—approximately 25% of the total cycles executed for *sendto()*. Secondly, implementing high-level, general-purpose interfaces in the operating system leads to inefficiency—the central argument used by proponents of vertical operating systems.

## 5.2 Concurrent Data Throughput

Two metrics are of prime importance when evaluating the suitability of an operating system for a network appliance—data transfer latency and peak transfer throughput. A particular concern for multiprocessor systems is how effectively the operating system supports concurrent, independent data transfers. In order to evaluate how effectively the Piglet architecture can support high-throughput data transfers we conducted a simple measurement of network transfer throughput as follows:

- A quad-processor 166MHz Pentium machine (*horsemen*) running the Piglet kernel was connected to both a dual-CPU 200MHz Pentium Pro (*logos*) and a dual-CPU 400MHz Pentium II (*moocow*), over 100Mb/s Fast Ethernet links (see Figure 5.5).
- The **ttcp** benchmark program was used to measure the mean aggregate bandwidth

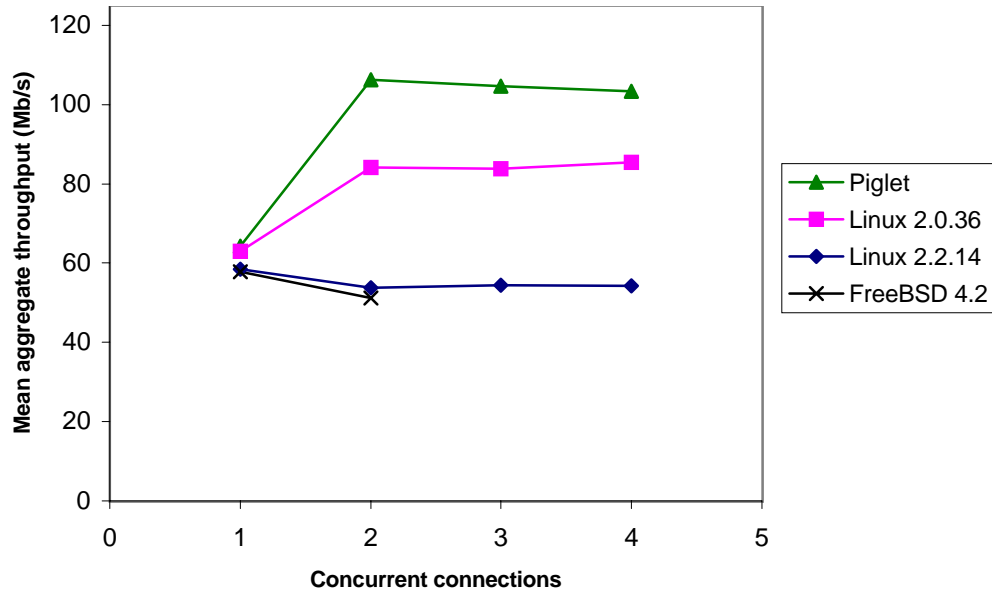


Figure 5.6: Mean aggregate throughput for  $n$  concurrent TCP connections

achievable by multiple concurrent TCP connections sending data from *horsemen* to both *moocow* and *logos*. The same, default parameters were used in all test configurations.

- Even-numbered connections were from *horsemen* to *moocow*, odd-numbered from *horsemen* to *logos* e.g., with 3 connections two are made to *moocow*, one to *logos*.

For the purposes of this experiment the Piglet user-space library was extended with a TCP protocol stack. While one benefit of the Piglet API is that it enables single-copy network operations, this feature was not used in this experiment; doing so would have required restructuring `ttcp`, and also would introduce an extra variable into the experimental setup, thus making it harder to isolate the cause of observed effects.

The mean (over 3 trials) aggregate throughput measured is plotted in Figure 5.6. For purposes of comparison the same measurements were also taken using various alternative operating systems on *horsemen*: two versions of the Linux kernel, 2.0.36 and 2.2.14, and FreeBSD 4.2. Both *moocow* and *logos* were running Linux kernel version 2.2.14. Unfortunately, other operating systems which one might wish to compare Piglet with could not be

evaluated due to either lack of SMP support (*NetBSD*, *OpenBSD*, *Nemesis*, *Exokernel*) or the failure of the software to install itself correctly on the test machine (*Solaris*).

### 5.2.1 Application Throughput Analysis

What we see from the graph is that Piglet makes better use of multiple processors than either version of Linux or FreeBSD. While all three systems are able to support approximately the same bandwidth,  $60\text{Mb/s}$ , over a single connection, Piglet can support approximately  $105\text{Mb/s}$  with multiple connections while Linux 2.0.36 and 2.2.14 are restricted to  $85\text{Mb/s}$  and  $55\text{Mb/s}$  respectively. FreeBSD appeared to have a bug in the network protocol stack which rendered it unable to complete the test for more than two concurrent connections—when the first connection to a given machine terminated the other connection was aborted due to an ‘I/O error’. However, the results for two connections suggest that its performance would be, at best, on a par with Linux.

An obvious concern for the other three systems tested is the lack of scalability beyond two concurrent connections. Let us address each system in turn.

1. *Linux 2.0.36* performed as expected. Two or more concurrent connections achieved higher aggregate throughput simply by exploiting idle time present when only a single connection is used. However, the use of a single global kernel lock prevents effective use of multiple processors on I/O bound tasks since only one process can enter the kernel at any time.
2. *Linux 2.2.14* surprised us with its very poor performance. One of the claimed advantages of the Linux 2.2 kernels over the 2.0 series is better utilisation of multiprocessors through finer-grained kernel concurrency control. However, our results clearly show that this is not true for I/O bound tasks. While we suspect that the ‘new and improved’ concurrency control is still the biggest problem, another factor is the more complex TCP/IP implementation, since we still observed 100% CPU utilisation even though the observed throughput was lower, even after disabling some of the advanced features supported by the protocol stack (specifically the use of the TCP timestamp option).

3. *Piglet* scales very well from one to two connections, mainly due to the TCP/IP stack being implemented in user-space, without a global lock, thus preventing serialisation which occurs in Linux. The performance for 3 connections was expected to be higher, but measurements taken for the next section show that the *Piglet* kernel is fully utilised for two concurrent connections using different network devices, so is unable to provide service to additional clients. There may also be some contention for the Linux global kernel lock since the clients are still scheduled and blocked by Linux.

Two positive results emerge from this experiment: the *Piglet* user-space protocol stack, unoptimised and without utilising the *Piglet* single-copy API, performs comparably—slightly better actually—to the Linux kernel protocol stack, and having per-client instances of the network subsystem in user-space, thus obviating the need for global concurrency control, provides the desired scalability. In order to better understand the polling behaviour of the *Piglet* kernel, however, it is necessary to investigate various characteristics of the *main* function.

Before doing so it is important to summarise how different design decisions incorporated in *Piglet* provide performance benefits. The `ping` latency measurements show the effect of replacing the Linux kernel protocol stack with the *Piglet* user-space protocol stack. This combines two factors: elimination of the system call overhead and removal of the multiple layers of abstraction present in the socket interface. Consider only packet transmission: the subsequent trace-based analysis shows that of the 3600 cycle ( $18\mu s$ ) difference approximately 1400 cycles of latency (`begin_send`–`Esys_sendto` + `Lsys_sendto`–`end_send`) are imposed by the system call mechanism.

The `ttcp` experiment demonstrates a significant benefit of moving the protocol stack to user-space, namely the improved scalability due to not having inter-network flow lock contention on the protocol state. However, neither experiment indicates how much benefit is derived from asynchrony or implicit concurrency in the *Piglet* system—this question is to be quantitatively addressed in future work. Measurements gathered in the following section, however, indicate that significant work is performed by the *Piglet* kernel in order to transmit packets, thus implying a reasonable degree of concurrency.



Operation	Mean processing time/ $\mu s$
GET_MEM	46.6
GET_FSET	24.2
CATCH_EVENT	23.3
SET_PEER	28.8
FREE_RX	3.7

Table 5.2: Mean processing time for various PSRs

### 5.3 Profiling the Polling Function

The biggest question raised by the adoption of the *Active Kernel* model is whether a polling model can perform acceptably when compared to conventional ‘passive’ operating system kernels. In order to address this question several properties of the *main* function have been investigated:

- How long the kernel takes to process various different *Posted Service Requests*.
- What is the overhead of polling shared-memory framesets to check for frames to be transmitted?
- How long does it take to transmit a frame?
- What overhead is imposed by polling devices rather than using interrupts?
- What fraction of the CPU is utilised by various workloads?

Each of these properties was evaluated using the same experimental configuration used in Section 5.2. While far from being an exhaustive list of metrics which can be used to characterise *main* this list offers useful, representative data.

#### 5.3.1 PSR Processing Time

The prototype Piglet implementation only uses posted service requests to provide a small number of operations, specifically those associated with the creation and manipulation of framesets. The mean processing time for each of these operations is shown in Table 5.2.

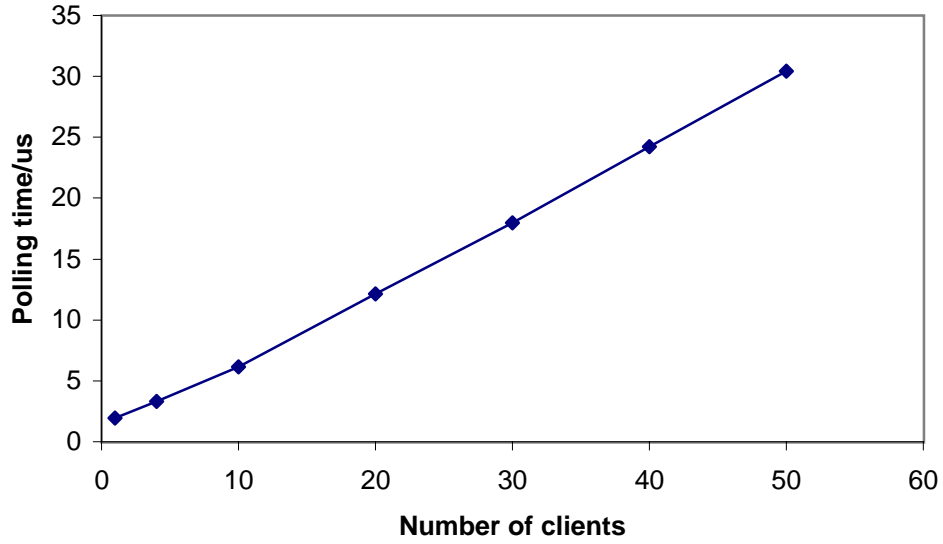


Figure 5.7: Frameset polling time as a function of number of clients

The purpose of each operation is as follows: **GET\_MEM** is used to allocate memory blocks from which a frameset is constructed; **GET\_FSET** performs this construction. **CATCH\_EVENT** is subsequently used to request that asynchronous signals be sent upon the occurrence of the ‘Packet Received’ event, and **SET\_PEER** associates routing information with the frameset. Each of these operations can be seen from the table to have a fairly high cost, however the nature of them as initialisation primitives means that this cost is typically only incurred once per frameset.

**FREE\_RX**, on the other hand, is called by the client for every packet received and enqueued on the receive queue, the purpose being to indicate to Piglet that the buffer has been processed and can be reclaimed for subsequent reuse. The overhead of  $3.7\mu s$  is sufficiently low that calling this function frequently does not impose undue load upon the kernel.

### 5.3.2 Frameset Polling Overhead

In an active kernel it is important that the overhead of polling not be so high as to render the system performance unacceptable. Two problems can occur if this is not the case: the

Pending frames	Frequency	Total Processing Time/ $\mu s$
0	94.0%	3.3
1	2.8%	32.9
2	0.3%	93.9
3	2.8%	135.0
4	0.1%	128.0

Table 5.3: Total frameset processing time for  $n$  pending frames

kernel spends all of its time polling, rather than performing services on behalf of clients; and the responsiveness of the kernel becomes poor, since the mean delay before a given object is polled is directly affected by the amount of time spent polling.

Figure 5.7 shows the mean time taken by the Piglet kernel to poll  $n$  user-space client framesets when no processing was required for any of the framesets. This represents the best-case overhead i.e., Piglet must spend at least this much time every polling cycle in order to examine  $n$  framesets. As expected, the polling time is essentially proportional to the number of framesets—the gradient of the line is approximately  $0.6\mu s$  per frameset.

### 5.3.3 Frame Transmission Cost

If one or more polled framesets has pending frames to be transmitted Piglet immediately passes them to the network device driver. If the cost of this processing is added to the cost of polling all framesets, measured above, then the resulting total represents the total amount of time the kernel spends processing user-space transmit queues per *main* cycle.

This total processing time was measured for a kernel handling four concurrent `ttcp` applications all sending to the same host. Every time the kernel polls all four framesets the number of pending frames can be anywhere from 0 to 4. As shown in Table 5.3 this total processing time can be fairly significant, especially when more than a single frameset has a pending frame. Thus Piglet can also be configured to only process one pending frame in each polling operation. However, it is worth noting that 94% of the polling operations find zero frames pending, leading to a mean total processing time which is calculated to be  $8.2\mu s$ .

Device state	Frequency	Processing Time/ $\mu s$
Idle	68.6%	3.9
Up-Complete	5.0%	45.0
Down-Complete	18.0%	16.4
Both	8.4%	48.1

Table 5.4: Device event handling time in different states

### 5.3.4 Device Polling Overhead

Similar consideration must also be given to the behaviour of the kernel when polling peripheral devices. One of the modules executed by *main* every cycle is responsible for polling every network device to determine when the device must be serviced, as would normally be indicated by an interrupt. In the prototype implementation this module checks three conditions: if a ‘downward’ (transmit) DMA operation completed, if an ‘upward’ DMA completed, and whether the statistics registers need to be read. Since the last of these occurs relatively infrequently only the performance when handling the first two conditions has been measured.

Table 5.4 shows the module’s processing time when the device is in each of four states: idle i.e., neither condition detected, one or the other DMA completed, or both. Similarly to the frame processing times, the idle processing time is relatively small as expected, but the other states have significant handling costs. In particular, processing of an *Up-Complete* condition involves passing the received packet upwards through every part of the network subsystem, hence the much higher processing time than the *Down-Complete* condition, which just requires that the driver update internal state. The expected handling time calculated from these results is  $11.9\mu s$ .

These results were generated using a single instance of **ttcp** as the kernel client; one side-effect of doing so is that a given condition being detected can indicate multiple instances of the associated event occurring e.g., multiple received packets transferred by DMA into memory. Unfortunately the number of event instances handled for each module cycle was not recorded, so instead the same statistics were gathered using **ping** as the client. Since **ping** only sends a single packet at a time and waits for a reply before

Application	CPU Utilisation	Idle Polling Period/ $\mu s$	Throughput/ $Mb/s$
<b>ttcp</b> $\times 1$	35%	9.0	64.3
<b>ttcp</b> $\times 2$	45%	10.5	65.6
<b>ttcp</b> $\times 3$	51%	11.0	68.8
<b>ttcp</b> $\times 2^a$	91%	15.0	105.0
<b>ping</b>	45%	9.0	N/A

<sup>a</sup>2 remote hosts

Table 5.5: CPU utilisation and polling period

sending another, there can be no instances of either both conditions being true or multiple events within a single condition. In this case the handling times for *Up-Complete* and *Down-Complete* were measured as  $33.8\mu s$  and  $8.0\mu s$  respectively.

### 5.3.5 CPU Utilisation

Finally, various measurements were made of the time consumed by the *main* polling function. The two metrics which are most interesting are the degree to which the kernel processor was utilised i.e., what fraction of the CPU cycles are spent in ‘useful’ work, and the *polling period*—the total execution time for one cycle of *main*. While this obviously depends on which tasks are performed by the modules called by *main*, the value which is perhaps most useful is the *idle polling period* i.e., the amount of time the kernel requires to perform a single polling cycle, since this dictates the responsiveness of the kernel to new events.

The utilisation and polling period were measured using both **ttcp** and **ping**, since each uses different packet sizes—**ttcp** sends MTU-sized packets, while the default **ping** invocation sends very small packets. In practice this results in **ping** sending packets at a faster rate than **ttcp**, thus requiring more processing by the kernel; since the kernel never touches packet payloads the larger size of the **ttcp** packets has no impact. The results from these measurements are shown in Table 5.5.

These results go some way toward explaining why Piglet’s aggregate **ttcp** throughput does not increase for  $> 2$  processes in the benchmark described in Section 5.2. Although the CPU utilisation is relatively high for a single connection, it does not increase greatly for 2 or 3 connections to the same machine, apparently because some other factor is

constraining the output—hence the data throughput not increasing. On the other hand, once two connections to distinct remote hosts are created the throughput does increase in a favourable manner; unfortunately the increase in CPU utilisation is greater than one might expect.

The final observation to be made about the idle polling period is that it is comparable to the overhead associated with an asynchronous interrupt in Linux, as measured in Section 2.7. Thus, at least for small numbers of clients, polling appears to offer similar event response times to interrupts—one concern of an active kernel system.

### 5.3.6 Analysis of Polling Function

The results garnered from these experiments allow several important design decisions to be evaluated. For each object which is polled by the Piglet kernel there are two criteria with which to determine whether polling is a viable communication mechanism: the best-case (idle) polling time and the processing time when an event must be processed. The best-case polling time must be low enough such that polling this object does not contribute a considerable amount to the polling period of *main*, thus increasing the kernel's response time. Similarly, the processing time for an object must not be so high as to render the kernel unresponsive while processing events occurring on that object.

Before analysing these results further it is important to note two factors which cast a more favourable light on them than the raw measurements. Firstly, the prototype Piglet kernel has not been optimised in any way, but the polling mechanism should be particularly amenable to certain kinds of optimisation e.g., run-time generation of polling functions. A similar idea was used in Massalin's *Synthesis* kernel [47]; see Section 8.4.1 for further discussion on this topic. Secondly, these measurements were obtained using very old hardware—a quad-166MHz Pentium Hewlett-Packard *Netserver ES*. It is entirely reasonable to assume that since CPU clock speeds are now almost an order of magnitude higher there would be a favourable decrease in measured polling times.

Now consider the results shown above for framesets and devices. The per-frameset polling time is sufficiently low that polling in conjunction with shared-memory objects appears to be a reasonable mechanism for communication with applications, if the number

of framesets which must be polled can be constrained within a relatively small upper-bound. Various kernel structural changes could be made to do so, these are discussed in Future Work. The processing time, however, is larger than might be deemed reasonable to guarantee acceptable response time to other objects e.g., devices. Note that the processing time itself is only an issue if the kernel does not respond to any other events during this time, a problem which can be alleviated by dividing the processing function into sub-functions which are interleaved with polling of high-priority objects.

Device polling, on the other hand, is more problematic to evaluate. The best-case polling time of  $3\mu s$  is sufficiently large to contribute a significant amount to the idle polling period of  $9\mu s$  for a single frameset. This suggests that polling objects with high best-case polling times may be too inefficient—perhaps interrupts should be used to signal when an event occurs but all processing should be done in a *schedulable region*, the approach used in the *Nemesis* kernel. Such a design retains much of the simplicity of the interrupt-free design but eliminates polling for expensive objects.

## Chapter 6

# Application-Level Performance: The Flash Web Server

Although microbenchmarks are important in analysing particular aspects of system performance they paint only a partial picture. In order to gain a better understanding of how well a particular operating system can support real applications it is necessary to run those applications on that operating system.

To evaluate the benefit Piglet provides to applications one must first select an appropriate application. Since Piglet is targeted at the network appliance application domain it is important to use a representative application—one whose workload consists primarily of data transfer operations rather than computation. The example used to evaluate Piglet is a web server, Vivek Pai’s *Flash* [69] server. *Flash* was deemed to be most suitable for two reasons: simplicity and performance. While *Apache* [1] is the dominant open-source web server it is reputed to be extremely complex and bloated, so was avoided by the author for practical reasons. *Flash*, on the other hand, is relatively simple and boasts excellent performance.

### 6.1 Porting Flash to Piglet

There are several ways in which *Flash* can be evaluated on top of Piglet. At one extreme of this spectrum is the use of a heavily-modified implementation of *Flash*, tailored to



use Piglet’s API and Piglet-specific features such as asynchronous service invocation and single-copy network transfers. While offering the highest potential performance such a system is more difficult to compare to Flash running on a conventional operating system since there are many more variables than just the operating system. Thus the opposite approach was adopted instead—evaluate Flash by providing a wrapper library to translate conventional operating system calls into Piglet requests, requiring absolutely no changes be made to Flash itself.

The set of functions used by Flash which must be translated to Piglet service requests includes all the network-related system calls. Unfortunately the Linux kernel, like all UNIX-based operating system, uses the same functions for network data transfer as for other forms of transfer e.g., file I/O, interprocess communication. Thus the library must differentiate all calls to such functions into the two distinct classes, translating network functions into Piglet requests but passing all other requests to the kernel. This extra layer of indirection, while alleviating the need to modify the application, imposes some overhead on passed-through system calls; the wrapper library is sufficiently simple, however, that this overhead is not likely to be significant.

Having previously designed and constructed such a user-space TCP library [19] it was with some trepidation that such a project was undertaken again. However, the focusing of Piglet toward the network appliance domain greatly simplifies certain aspects of the implementation:

1. *Predictability of execution:* applications for network appliances have a much more homogeneous structure which can be taken advantage of in the library implementation.
2. *Asynchronous communication:* network appliances make heavy use of asynchronous (non-blocking in the UNIX parlance) I/O operations, which map more readily onto Piglet functions.

The single biggest problem encountered in the earlier TCP stack implementation was designing the library to deal with arbitrary applications which may make only sporadic use

of the library. Many of the periodic functions required for protocol operation and book-keeping must thus arrange to be executed without the application invoking the library to do so, instead using, say, an interval timer callback or additional process/thread. This adds significant complexity in order to maintain data consistency in the face of these unsynchronised functions.

If the Piglet library is restricted to network appliance applications then the common structure of server applications can be used to eliminate this complexity. Most servers are structured around a central event-dispatch function which determines the set of operations which should be performed at the current time and executes the appropriate functions. This determination of events to handle is usually accomplished using the *select* system call—an application calls *select* with a set of objects e.g., network connections, it wishes to perform some operation upon, and *select* indicates which operations can proceed without the application blocking. This determination of object readiness and processing of events constitutes the core of most servers; since the application repeatedly calls the same function this provides a natural point at which the library can perform book-keeping and other periodic operations. Thus the requirement for complex asynchronous processing is eliminated.

The second advantage of constructing a TCP library for server applications is the use of asynchronous rather than synchronous network I/O operations. While general-purpose applications usually use the more straightforward synchronous system calls, since the application is notified immediately of the success or failure of the call, servers tend to use asynchronous operations in order to maximise throughput. This means that they can be easily mapped onto Piglet's asynchronous API without requiring significant complexity in the library.

The final complication in implementing the user-space TCP stack arose from the nature of *select*. Because Flash uses *select* to check for both network connection events (e.g., a HTTP request being received or transmit buffer being drained) and interprocess communication, such as a signal from a file cache helper, the library has to be able to handle both Piglet objects—network connections—and kernel structures in a single call. This is accomplished by first checking the Piglet objects, removing those from the query set, then

File	Size	Frequency
file500.html	500 bytes	35%
file5k.html	5kB	50%
file50k.html	50kB	14%
file500k.html	500kB	0.9%
file5m.html	5MB	0.1%

Table 6.1: Distribution of file sizes for Webstone benchmark

Test System	Server CPUs	Server Throughput/ <i>Mb/s</i>	Connections per second
Piglet	1	71.34	438.2
Linux 2.2.16	1	45.5	281.7
	2	43.5	273.6
	3	42.8	268.4
	4	42.1	262.2

Table 6.2: Flash performance for the Webstone benchmark

calling the Linux *select* function to obtain the results for the kernel objects. However, before doing so the ‘timeout’ parameter must be updated to reflect the results of querying the Piglet objects: if any events were already pending then the Linux call must not block. Once that call returns the results must be merged with those for the Piglet objects, which may need to be rechecked if none had pending events before the call.

## 6.2 Evaluating Flash’s Performance

Flash was evaluated by running it on the same testbed described in the previous chapter. The server itself was run on the *horsemen* quad-CPU machine, running under either Piglet, Linux-2.2.16, or FreeBSD 4.2, with default parameters and the number of server processes varying from one up to the maximum number of available application CPUs—3 for Piglet, 4 for Linux and FreeBSD.

Two different programs were used on the client machines to generate requests for the server: the *Webstone 2.5* [55] web benchmark program, and the *curl* [84] utility for automated i.e., non-interactive, retrieval of web pages from a server

Test System	Server CPUs	Server Throughput/ <i>Mb/s</i>
Piglet	1	75.0
	2	93.7
	3	98.8
Linux 2.2.16	1	50.83
	2	51.14
	3	50.86
	4	51.38
FreeBSD 4.2	1	51.30
	2	51.24
	3	32.0
	4	24.0

Table 6.3: Flash performance for the *curl* test

Webstone allows the user to specify a page-size distribution which the clients use to determine which pages to fetch from the server. The distribution shown in Table 6.1 is supposedly representative of ‘real’ web traffic. A coordinating process starts a number of clients on each client machine (5 per machine in this case) and provides them with the necessary parameters to drive the server. After a fixed time interval, one minute for this test, the coordinator queries the clients for various traffic statistics and produces overall server performance numbers. These results are shown in Table 6.2.

Unfortunately the prototype Piglet implementation was not stable enough to support Flash running with more than a single server process when driven by Webstone. In order to provide a comparison with Linux *curl* was used to generate upper-bounds for server performance by repeatedly downloading the largest file in the distribution (5Mb) and measuring the average throughput. These measurements are shown in Table 6.3; these numbers are essentially the same as those obtained using the **ttcp** benchmark earlier.

These throughput figures were used to calculate projected values for the Webstone benchmark on Piglet by multiplying the *curl* throughput for 2 or 3 server CPUs by the ratio between the Webstone and *curl* throughput for a single CPU—95%. Both sets of throughput measurements are displayed in Figure 6.1.

Similarly, when FreeBSD was used as the test platform for the WebStone test all web

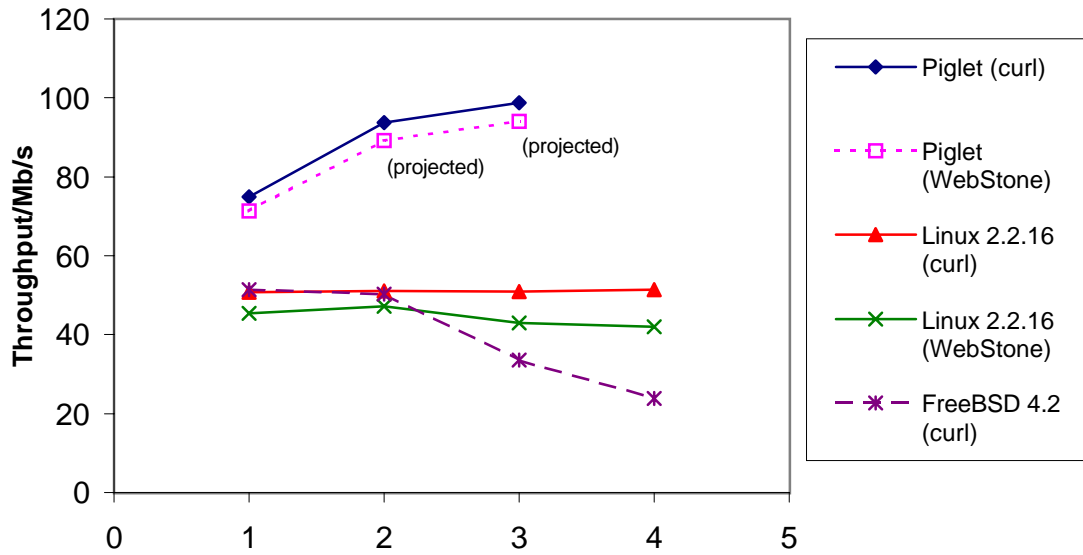


Figure 6.1: Flash performance for  $n$  server processes

connections would cease transmitting after a few seconds, regardless of the number of server processes. Fortunately it proved possible to at least complete the *curl* measurements on this platform. However, the results show unexpectedly poor performance for more than two server processes; one might surmise that perhaps FreeBSD is optimised for the common case of dual-processor SMP systems in a way which penalises systems with a higher degree of multiprocessing.

What we see from this graph is that Piglet and Linux both exhibit the same scalability behaviour as was observed with *ttcp*. Piglet shows some throughput increase from 1 to 2 CPUs, but beyond that the kernel CPU is over-utilised and thus restricts the maximum throughput achievable. Linux is again disappointing, showing zero or negative dependence upon the number of server processors.

## Chapter 7

# Related Work

There has been a significant amount of previous operating systems research which has relevance for Piglet. Both the problem of intrusion and new system architectures to address that problem have long been major directions in systems research. More recently, there have been a number of projects in addition to Piglet which are targeted specifically at network appliances.

### 7.1 Intrusion

While the generalised concept of operating system intrusion is one of the contributions of this work, both policy and mechanism intrusion have been investigated to some degree by previous researchers. The problem of policy intrusion has been known of at least since the early '80s, and specific examples of mechanism intrusion have been studied by various groups. In response, a number of operating system projects which attempt to reduce policy intrusion, and to a lesser extent mechanism intrusion, have been designed and implemented.

Perhaps the clearest demonstration of the significant negative impact which inappropriate operating system structures can have on application performance was provided by Stonebraker's discussion [85] of how the services provided by a general-purpose OS (UNIX) are unsatisfactory for supporting a high-performance database management system (DBMS). He comes to the conclusion that "A DBMS would prefer a small efficient operating system with only desired services... On the other hand, most general-purpose

operating systems offer all things to all people at much higher overhead.”

While Stonebraker implies that the needs of any given application can be most efficiently met by a special-purpose operating system, a general-purpose OS which offers low degrees of both policy and mechanism intrusion may also be able to do so. Hence the need for alternative operating system architectures, specifically those which facilitate application-specific resource management with low overhead.

### 7.1.1 Policy Intrusion

Stonebraker’s analysis of the limitations imposed by a general-purpose operating system upon a database management system provides many examples of policy intrusion: inappropriate buffer replacement policies in the file system buffers, inefficient physical layout of data upon disks, high overhead of general-purpose scheduling and process management. Common to every example is the problem of the abstractions implemented by the OS being restricted to policies which interact adversely with those used by the DBMS. Hence, the need to separate resource management mechanisms and policies is clearly demonstrated for many different classes of resource.

Application-specific management of virtual memory has been extensively researched: Appel and Li [2] provide general considerations for implementing application-level VM primitives; Harty and Cheriton [30] discuss an implementation of application-controlled physical memory in the *V++* system and its benefits in a transaction processing application; and finally, Hand [28] and Engler [21] describe specific implementations for vertically-structured operating systems. Hand discusses how *self-paging* is used in the Nemesis OS to support virtual memory without compromising the QoS guarantees of other applications, while Engler describes how the low-level primitives exported by the *Aegis* exokernel can be used to efficiently support various application-level functions.

User-level network protocols are another area which has been covered by many groups: Cornell’s *U-Net* [90], and the *Virtual Interface Architecture* [13] derived from it, provide applications with a direct interface to the network adapter; Thekkath et al. [87], and Edwards and Muir [19] describe user-level implementations of network protocols. The

flexibility to implement application-specific network protocols and/or management of network resources has become much more important for both multiservice operating systems and high-performance network appliances.

### 7.1.2 Mechanism Intrusion

While the need for application-specific resource management appears to have been universally adopted, the reduction of mechanism intrusion is often taken to be unimportant. For example, Engler et al. describe in [21] how Aegis, the first implementation of their exokernel architecture, was heavily optimised to reduce the costs of invoking low-level primitives. Subsequently, Kaashoek et al. reported in [39] that such optimisations are not necessary to leverage the most benefit from the exokernel architecture. While this statement is accepted as generally true, in certain classes of application e.g., where I/O comprises the bulk of the workload, it appears that there is considerable benefit to be had from reducing mechanism intrusion.

Mogul and Ramakrishnan [59] describe how a conventional operating system can become ‘livelocked’ due to continuously being interrupted by a network interface card. While the solution they propose, temporarily disabling interrupts and switching to a polling mechanism, is very similar to the Piglet architecture it differs significantly in that their kernel switches dynamically between the two schemes while Piglet always uses polling. Smith and Traw [82] introduced *clocked interrupts* as an alternative combination of interrupts and polling. They propose having a periodic timer interrupt which polls devices, effectively ‘clocking’ interrupts into the system at a controlled rate. In light of the analysis of the polling function presented earlier such hybrid schemes deserve further investigation.

Dougan et al. [18] discuss possible modifications to the Linux kernel in order to reduce the impact of the OS on cache performance. In particular, they recommend bypassing the cache in certain portions of the kernel e.g., when zeroing memory pages in the idle task, so as not to pollute the cache.

One of the most prominent examples of mechanism intrusion is the high cost of inter-process communication in the early microkernel architectures, as described below. Subsequently, several OS projects made significant efforts to reduce this cost e.g., the L3 and



L4 microkernels [29, 43], the Spring microkernel [26]. These systems are described in more detail below.

### 7.1.3 Multiprocessor Intrusion

The cost of synchronisation in a multiprocessor system was noted earlier as being an instance of mechanism intrusion specific to such architectures. Of particular relevance to Piglet are a number of projects in the early 90's which examined software architectures for applying multiple processors to network protocol processing.

Björkman and Gunningberg [7] modified the *x-kernel* [34] to run on a 26-processor Sequent Symmetry as a platform for investigating different parallel implementations of TCP and UDP stacks. Their method, having each packet processed by a different processor, demonstrated good speedups: proportional to the number of processors for UDP, but much less for TCP.

Nahum et al. [65] performed a very similar experiment, also using the *x-kernel* as their basis for a processor-per-packet parallel implementation of various protocols. They too found that simple protocols scale well with number of processors but in more complex ones the locking required by shared state severely hampers concurrency. One particularly interesting result presented by Nahum's group is the observation that complex locking schemes, with multiple locks for different components of the protocol stack, resulted in lower overall performance than a simple model with a single lock for the whole stack.

Of direct relevance to Piglet is the conclusion of both groups that contention for shared resources is the primary limiting factor for simple protocols such as UDP, but synchronisation dominates for more complex protocols. This suggests that an approach which minimises the amount of state shared between network flows, such as the user-space protocol implementations in Piglet, is an important step in maximising the benefit of multiple processors. This conclusion is echoed by Nahum et al., who state that intra-flow parallelism is severely limited by the locking which is required, while inter-flow parallelism is reasonably scalable.

An alternative use of multiple processing elements in a network protocol stack is to implement some or all of the stack functionality on the network interface itself. Cooper et

al. designed the *Nectar* communication processor [14] in such a manner as the host interface for systems connected to the Nectar network. Each *communications accelerator board* (*CAB*) contains a CPU and local memory which are used to run a specialised operating system; the relationship between the CAB operating system and host is similar to that between Piglet and Linux in the hybrid prototype. The most interesting result from this project is the increase in throughput when the TCP stack was moved from the host operating system onto the CAB, approximately a factor of 3. However, since the authors do not describe the implementation details of this experiment it is difficult to deduce whether such a relocation of functionality would be profitable in Piglet.

## 7.2 Alternative Operating System Architectures

Many different system architectures have been designed to reduce both policy and mechanism intrusion. The microkernel architecture was perhaps the first to be designed with application-specific resource management as a primary goal. The high cost of inter-process communication (IPC) in such systems led to three different subsequent directions in operating systems research:

- *Fast IPC*-based microkernels, which retained the same architecture but attempted to reduce the cost of IPC.
- *Vertically-structured* operating systems, which eliminated that cost by moving functionality from servers into applications themselves.
- *Extensible* kernels, which permitted application-specific code to be added as extensions to the kernel.

Note that these directions are in fact orthogonal, and thus it is possible for an operating system to incorporate ideas from more than one, leading to a 3-dimensional design/feature space.

### 7.2.1 Microkernels

Early examples of the microkernel architecture include the Bell Labs *Multi-Environment Real-Time (MERT)* system [45] and the *Series/1 Distributed System (SODS)* [81, 27]. MERT was intended to support multiple system environments on a single physical machine (much like IBM’s VM/370 system [16]), specifically the combination of time-sharing (UNIX) and real-time operating systems. SODS/OS was designed as the building block for a distributed computing environment in which processes were designed to be location-independent, thus permitting transparent migration. Neither MERT nor SODS/OS was designed with efficient resource management as a primary goal.

Both MERT and SODS/OS conform closely to the characteristics of a microkernel architecture, namely provision of system functions by server processes, and the use of messages as the primary IPC mechanism. Like subsequent microkernel projects they both suffered from a high cost of communication; Lycklama [45] states that “the MERT system requires from 5 to 50 percent more system time for the more heavily used system calls”, while Hammond [27] observes that “the associated context swapping is extremely expensive on a conventional machine”.

*Mach* [77, 76] perhaps typifies the microkernel architecture. The kernel itself implements only a core set of services necessary to support application-level servers, which in turn provide the system services required by applications. A powerful message-passing mechanism provides the primary IPC mechanism; in practice the high degree of mechanism intrusion this introduced meant that the performance of applications running on top of server-based operating systems e.g., the 4.3BSD server, was typically much lower than the native environment.

The poor performance of Mach, primarily due to this high cost of IPC, prompted several research groups to address the problem of reducing this cost. Liedtke’s *L3* [43] and *L4* [29] microkernels left the fundamental architecture unchanged but used the minimisation of IPC cost as the overriding priority in the system design; Liedtke [43] reports IPC times more than an order of magnitude faster than Mach. Even so, *lmbench* [49] results presented by Hartig [29] show that Linux running natively is still faster than the Linux server running on L4.

Sun's *Spring* microkernel introduced the concept of 'doors' as a mechanism by which applications can perform control transfers between protection domains. A door is conceptually similar to a system call entry point in a conventional OS, in that it defines a fixed access point by which an untrusted client may invoke a specified function. However, doors are defined by applications and passed as capabilities to clients, thus permitting those clients to perform fast transfers of control to the server.

Both *Amoeba* [86] and *EROS* [80] used capabilities to control access to system objects. In both systems these capabilities are used as the primary service invocation mechanism—an application invokes a particular operation on a capability, which is either implemented directly by the kernel or passed to a server process using IPC. Clients need not be aware of this distinction, thus providing a degree of flexibility in system structure. Amoeba and EROS differ primarily in their target application domain: Amoeba was designed as a platform for distributed computing while EROS is intended to combine the flexibility of a general-purpose operating system with the security provided by capabilities.

The University of Massachusetts' *Spring* real-time microkernel [67] (not related to the aforementioned project of the same name) is perhaps most similar to Piglet in that each node in the Spring distributed system dedicates one of four CPUs to handling system tasks. The functionality of this processor is not described in the available documentation, other than to state that it is used to handle administrative tasks and shield applications from device interrupts. This is primarily done to reduce unpredictability in the system, thus assisting the scheduling of jobs with hard real-time requirements.

While Piglet appears to share a fundamental message-based programming model with many microkernel systems this similarity is only skin-deep. Message-based microkernels in fact implement message processing as synchronous IPC, thus translating the message-based interface into a conventional function call model. In particular, 'messages' are not buffered—instead the kernel blocks the 'sender' until the 'receiver' is willing to accept the message. Piglet, on the other hand, not only exports a message-based API but implements it using 'real' messages, resulting in true asynchronous communication.

### 7.2.2 Vertically-Structured Operating Systems

While fast IPC mechanisms have helped lessen the biggest problem with the microkernel architecture, namely the high overhead of communicating with servers, others still exist e.g., QoS crosstalk. Hence the development of the vertically-structured operating system as an alternative which does not share the same architectural limitations. The University of Cambridge's *Nemesis* kernel [42] and MIT's *Exokernel* architecture [21, 39] are probably the best-known examples of such operating systems, but other groups have also designed operating systems which loosely fit into the same model e.g., Stanford's *Cache kernel* [11].

One of the precursors to vertically-structured operating systems which proposed moving away from the microkernel approach was the *Swift* system [12]. The Swift architects advocated for the use of *upcalls* as the fundamental communication mechanism between modules—synchronous procedure call rather than asynchronous IPC between tasks, primarily motivated by the overhead of IPC and complexity introduced by asynchrony. The converse decision was made in the Piglet design—the active kernel provides a much lower overhead channel than conventional IPC, and the complexity of asynchronous communication is considered a small price to pay in return for the implicit concurrency derived from it.

Nemesis exemplifies the vertically-structured operating system architecture. The kernel exports only the minimum set of functions necessary to support application-level resource management—these are identified by Barham [5] as translation, protection and multiplexing. Nemesis separates control- and data-plane operations—the former e.g. acquiring access to a region of disk blocks, are typically performed by server tasks, while the latter e.g., reading/writing those disk blocks, are optimised so as to require little or no interaction with the kernel.

MIT's exokernel architecture is very similar to Nemesis, sharing many of the key features i.e., exporting low-level resource management primitives, optimising data-plane operations to reduce kernel interaction. Where it differs from Nemesis is in its use of kernel extensibility to minimise the cost of the most common kernel operations. This concept is described in more detail below.

The Cache kernel also provides applications with a set of low-level primitives which

support application-level resource management. However, rather than implementing protection and multiplexing, the Cache kernel instead exports a minimal core set of objects (address spaces, threads, and kernels) upon which applications build services.

### 7.2.3 Extensible Operating Systems

While vertically-structured operating systems provide application-specific resource management by exporting low-level interfaces which applications can use to implement their own policies, an alternative approach is to allow applications to extend the OS kernel with functions which implement those policies. This principle is embodied in the University of Washington's SPIN kernel [6] and, to a lesser degree, MIT's exokernel and the Synthesis OS [47].

The SPIN system permits applications to construct kernel extensions using Modula-3 [66], a type-safe object-oriented language. These extensions take the form of functions executed in response to kernel events e.g., a packet being received from a network interface, and can contain almost arbitrary sequences of code. To maintain the integrity of the kernel certain restrictions are enforced e.g., interrupt handlers must be written in such a manner that the kernel can safely terminate the function if it executes for too long. Thus, by exposing low-level kernel events to applications and allowing them to execute specific functions in response SPIN permits applications to embed their own resource-management interfaces in the kernel.

The exokernel uses a more restricted form of kernel extension. Rather than allowing applications to directly download arbitrary functions into the kernel, the exokernel instead allows applications to specify in a function-specific manner the behaviour of certain functions. For example, restricted languages are used to generate packet filter functions in order to demultiplex received network packets; similarly, the *XN* storage manager uses application-specified *untrusted deterministic functions (UDFs)* to manipulate metadata without the kernel having to be aware of the metadata format.

Finally, much like the exokernel although in fact preceding it, Massalin's Synthesis OS [47] made extensive use of run-time code generation to construct specialised kernel functions. These functions were used for various tasks, including scheduling, interrupt

handling, and system call invocation. Synthesis differs primarily from SPIN and the exokernel in that extension is controlled wholly by the kernel itself and not by applications. Thus Synthesis is perhaps most interesting in that it reduces mechanism intrusion but not policy intrusion—the converse approach to that taken by most other contemporary operating systems.

### 7.3 Operating Systems for Network Appliances

The operating system architectures described above were originally designed to be general-purpose systems, but many have been appropriated for use in special-purpose applications e.g., Menage’s *Rcane* platform [51] for supporting active network services uses Nemesis as the base operating system. There are also a number of operating systems which, like Piglet, have been designed specifically with network appliances in mind.

Ramakrishnan et al. designed and implemented a small real-time kernel as a platform for a video-on-demand file server [75]. Their system is as close to the definition of network appliance as possible—its only function is the transmission of video files in response to requests received from the network. One area of similarity to Piglet is the use of polling within device drivers for communication with network devices: this was observed to eliminate receive livelock and provide a degree of control over the amount of system resources consumed by network requests.

The *Scout* operating system [61] introduces the concept of *paths* as a means for specialising operating system structure for particular data flows within the system. A path defines a set of modules through which data flows and specifies particular semantic properties of that data flow. These properties are used by the Scout kernel to specialise and optimise the modules instantiated when a path is created. Scout has been used to construct various network appliances, including an MPEG decoding terminal [62] and an optimised TCP forwarder [83].

MIT’s *Click* router [41], while not an operating system but a router implemented on top of Linux, is also based around a modular design. Click allows users to statically construct a router from a set of standard components, then optimises the connections

between those modules. The most interesting aspect of Click in relation to Piglet is the use of a polling device driver, which the system designers estimate to provide a fourfold increase in forwarding rate for minimum-size packets. They also make a similar observation about the overhead of interrupt handling, estimating the cost on a 700MHz Pentium III as being  $4.8\mu s$  of the  $13\mu s$  required to forward a packet.

Donnelly's *eXpert* operating system combines paths with tasks, equivalent to processes in a conventional operating system. While paths are used for data processing, tasks are retained for those tasks which they are better suited to, such as background maintenance tasks which are not necessarily data-driven. *eXpert* is still under development but is targeted at exactly the same set of applications as Piglet.

It is also interesting to compare Piglet to an operating system designed expressly for a single network-appliance—Cisco's *IOS* [8], the operating system used in all Cisco routers. In particular, two of the design decisions taken by the IOS architects are the same ones taken for Piglet kernel services: there is no intra-kernel memory protection and services are scheduled using cooperative multitasking. The success of IOS offers some justification that these choices are reasonable ones.

## 7.4 Message-based Systems

Finally, it was stated above that Piglet differs from microkernel operating systems in its use of asynchronous, rather than synchronous, message passing for communication. It is worth briefly noting a number of systems which have also been based around asynchronous message passing.

IBM's *Enterprise Systems Architecture/390* mainframe [35], and its predecessors including the System/360 and System/370, uses *channels* as the means by which I/O devices are connected to the system. Channels are actually smaller computer systems to which the main CPUs delegate the task of performing I/O. This is accomplished by means of a series of *Channel Control Words*, analogous to Piglet's Posted Service Requests, which specify operations to be performed by the channel on some I/O device. Asynchronous processing of these operations by the channel provides the same concurrency and asynchrony



benefits that Piglet derives from PSRs.

Message passing has also been used as a fundamental communication mechanism in some large-scale multiprocessor computers. Prominent examples of such machines are the MIT/Intel *J-Machine* [17] (‘J’ for jellybean, since the CPUs are designed to be inexpensive and plentiful, like jellybeans) and the Berkeley *Active Messages* system [89]. The core of the J-Machine, the Message Driven Processor (MDP), was designed to incorporate efficient mechanisms for communication. Most relevant to Piglet is the addition of a special instruction to send a message directly from user-space, and efficient hardware dispatch of incoming messages to application handlers. This emphasis on reducing the application communication overhead is similar to the motivation for shared-object communication in Piglet, but using a network interconnect rather than shared memory.

The Active Message project was motivated by two problems observed in earlier message passing multiprocessor systems: poor overlap of communication and computation, and high communication overhead. These are two of the primary factors motivating Piglet, so one would expect there to be some common ground. Indeed, von Eicken et al. found that asynchronous communication enabled concurrency between message senders and receivers, analogous to clients and the Piglet kernel, and direct user-space access to communication mechanisms was important in reducing overhead.

## Chapter 8

# Conclusions and Future Research

The recent and continuing explosive growth of the Internet has created a requirement for inexpensive but high-performance *network appliances*—systems dedicated to applications whose primary function is the transfer of data rather than computation. This characteristic of application behaviour presents the operating system designer with a significant opportunity for specialisation and optimisation which a general-purpose operating system is unable to take advantage of. Hence the design and implementation of Piglet as a novel operating system targeted specifically at the network appliance domain.

This thesis presents several contributions in the realm of operating system design, the most notable being:

1. Classification and analysis of different forms of operating system intrusion.
2. Identification of those aspects of existing operating systems which contribute to intrusion.
3. Design of a new operating system architecture, the *Active Kernel*.
4. Implementation of the first instance of this architecture—*Piglet*.
5. Evaluation of the Piglet architecture using low-level benchmarks.
6. Porting and performance measurement of a typical network appliance application—the *Flash* web server.

Each of these claims will briefly be expanded upon in the following sections.

## 8.1 Analysing Operating System Intrusion

Operating systems perform two, somewhat orthogonal, functions: virtualisation of physical resources and provision of a library of common system functions e.g., I/O operations. Unfortunately these two functions are often unnecessarily conflated into a single entity—the operating system kernel. This leads to suboptimal application performance if the policies embedded within the library of system functions are not appropriate for a given application—*policy intrusion*.

Various operating system architectures have been proposed, including microkernels and vertically-structured systems, which successfully address this problem. However, another form of intrusion—*mechanism intrusion*—is inherent in the service invocation mechanisms used in these systems and those which they attempt to improve upon. Therefore a new operating system structure is required in order to address this latter problem.

## 8.2 Piglet—Reducing Intrusion by Activating the Kernel

This thesis proposes a new operating system structure which goes some way toward successfully tackling the problem of mechanism intrusion. This is accomplished by utilising alternative mechanisms for traversing privilege barriers to those used in conventional systems. The synchronous, procedure-call oriented mechanism which has been used in virtually all existing operating systems is abandoned in favour of an asynchronous, message-passing scheme using shared memory as the message channel. One processor in a multiprocessor system is dedicated to continuous execution of the *Active Kernel*, thus elevating the kernel from a passive shared library to an active entity.

Chapter 3 describes *Piglet*, a complete operating system design based upon the active kernel and targeted at the network appliance application domain. This design has been implemented in a hybrid Piglet-Linux system that permits applications to be created which exercise Piglet's capabilities while utilising the existing Linux kernel for those functions not provided by Piglet.

## 8.3 Performance of the Active Kernel

A significant contribution of this thesis is a demonstration of performance benefits realised by the Piglet architecture, and potential areas for further improvement. Microbenchmark experiments in Chapter 5 show that Piglet successfully increases the performance of applications compared to a conventional operating system; it provides both lower latency transmission for a given application and higher throughput of data for multiple concurrent applications.

Characterisation of the Piglet kernel's polling function shows that polling is indeed a viable communication mechanism for certain classes of objects, in particular shared-memory objects used by applications to post service requests. The results obtained for polling of I/O devices, however, suggest that a hybrid mechanism may be more appropriate in order to guarantee acceptable system responsiveness.

Finally, the porting of an existing web server application from a conventional operating system to the Piglet prototype was described. Even without exploiting any of Piglet's features intended specifically to boost the performance of such applications the overall performance of the web server when running on Piglet was substantially higher than when running on the original conventional operating system.

## 8.4 Future Work

While the results presented herein go some way toward proving the viability of the active kernel as an alternative to existing operating systems, further work must be carried out in order to confirm Piglet's suitability as a platform for network appliances.

### 8.4.1 Optimising the Active Kernel

One concern raised by the analysis of the *main* polling function is the scalability of polling as the basis of the Piglet kernel. Two observations must be made in order to make the system scalable: reduction of object polling and event processing time, and minimisation of the set of objects which must be polled.

One method for optimising polling and processing time which appears potentially fruitful is the use of run-time code generation, as demonstrated in the *Synthesis* kernel. Polling in particular, since it usually consists of repetitive calls to a simple function, should benefit from inline expansion and run-time constant propagation.

The set of objects which must be polled for any given iteration of *main* can be reduced by taking advantage of kernel state indicating which objects could be updated by active processes during that iteration. For example: on an  $n$ -CPU system, only  $n$  processes can be executing at any given time, so only the corresponding objects need be polled. Further reduction of the polled set can be achieved by multiplexing many application-level objects onto a single kernel polled entity e.g., many network endpoints which share QoS properties can all share a single frameset.

#### 8.4.2 Evaluation of Alternative Applications

Piglet must also be evaluated in more than a single application environment with the network appliance domain. Of particular interest are *forwarding appliances*, since recent trends in hardware design for such platforms are toward heterogeneous processing capabilities in a single system. For example, Intel's *IXP1200* network processor [36] places a *StrongARM* core on an ASIC with six *micromachines* for packet forwarding; several of these devices are then attached to the PCI bus of a standard PC host system to create a high-performance router. Such systems require simple but high-performance operating systems to execute on the control processor (StrongARM) which communicates with both host and forwarders—Piglet would appear to be an ideal candidate.

# Appendix A

## Linux ping Execution Trace

These measurements show the value of the (64-bit) timestamp counter recorded at the point of entry (Exxx) and exit (Lxxx) from each function executed in the transmit path of the Linux kernel ‘raw’ IP stack. They are represented graphically in Figure 5.3.

```
begin_pinger: 1328188896
begin_send: 1328190439
Esys_sendto: 1328191174
Einnet_sendmsg: 1328191857
Eraw_sendto: 1328192100
Eip_build_xmit: 1328192463
Eboomerang_tx: 1328194362
Lboomerang_tx: 1328195466
Lip_build_xmit: 1328195607
Lraw_sendto: 1328195709
Linnet_sendmsg: 1328195919
Lsys_sendto: 1328196051
end_send: 1328196717
Evortex_interrupt: 1328198649
vortex_down_complete: 1328199105
Lvortex_interrupt: 1328200575
begin_select: 1328212147
```

Evortex\_interrupt: 1328216275  
vortex\_up\_complete: 1328216717  
Enetif\_rx: 1328218164  
Lnetif\_rx: 1328218346  
Lvortex\_interrupt: 1328219074  
Enet\_bh: 1328220307  
Eip\_rcv: 1328220967  
Eicmp\_rcv: 1328222086  
Licmp\_rcv: 1328222720  
Lip\_rcv: 1328223124  
Lnet\_bh: 1328223209  
end\_select: 1328225838  
Esys\_recvfrom: 1328226744  
Einet\_recvmsg: 1328227308  
Eraw\_recvmsg: 1328227560  
Lraw\_recvmsg: 1328228682  
Linet\_recvmsg: 1328228841  
Lsys\_recvfrom: 1328229369  
got\_packet: 1328230030

# Bibliography

- [1] Apache Software Foundation. Apache HTTP Server. [www.apache.org/httpd.html](http://www.apache.org/httpd.html).
- [2] A. W. Appel and K. Li. Virtual memory primitives for user programs. In *Proc. of the 4th Int'l. Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 95–109, October 1991.
- [3] Various Artists. Squid Web Proxy Cache. [www.squid-cache.org/](http://www.squid-cache.org/).
- [4] G. Back, W. C. Hsieh, and J. Lepreau. Processes in KaffeOS: Isolation, resource management, and sharing in Java. In *Proc. of the 4th Symp. on Operating Systems Design and Implementation*, October 2000.
- [5] P. R. Barham. *Devices in a Multi-Service Operating System*. PhD thesis, University of Cambridge, July 1996.
- [6] B. Bershad et al. Extensibility, safety and performance in the SPIN operating system. In *Proc. of the 15th ACM Symp. on Operating Systems Principles*, pages 267–284, December 1995.
- [7] Mats Björkman and Per Gunningberg. Locking effects in multiprocessor implementation of protocols. In *Proc. of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 74–83, September 1993.
- [8] Vijay Bollapragada et al. *Inside Cisco IOS Software Architecture*. Cisco Press, 2000.
- [9] CERT Coordination Center. Advisory CA-2001-02, Multiple Vulnerabilities in BIND, January 2001. [www.cert.org/advisories/CA-2001-02.html](http://www.cert.org/advisories/CA-2001-02.html).



- [10] F. Chang and G. A. Gibson. Automatic I/O hint generation through speculative execution. In *Proc. of the 3rd Symp. on Operating Systems Design and Implementation*, pages 1–14, February 1999.
- [11] D. R. Cheriton and K. J. Duda. A caching model of operating system functionality. In *Proc. of the 1st Symp. on Operating Systems Design and Implementation*, pages 179–193, November 1994.
- [12] D. Clark. The structuring of systems using upcalls. *Proc. of the 10th ACM Symp. on Operating Systems Principles*, pages 171–180, 1985.
- [13] Compaq Computer Corp., Intel Corporation, Microsoft Corporation. *Virtual Interface Architecture Specification Version 1.0*, 1997. [www.viarch.org](http://www.viarch.org).
- [14] Eric C. Cooper et al. Protocol implementation in the nectar communication processor. In *Proc. of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 135–144, September 1990.
- [15] Fernando J. Corbató et al. An experimental time-sharing system. In *AFIPS Conference Proceedings (Spring Joint Computer Conference)*, volume 21, pages 335–344, May 1962.
- [16] R. J. Creasy. The origin of the VM/370 time-sharing system. *IBM Journal of Research and Development*, 25(5):483–490, September 1981.
- [17] W. J. Dally et al. The message-driven processor: A multicomputer processing node with efficient mechanisms. *IEEE Micro*, pages 23–39, April 1992.
- [18] C. Dougan et al. Optimizing the idle task and other MMU tricks. In *Proc. of the 3rd Symp. on Operating Systems Design and Implementation*, pages 229–236, February 1999.
- [19] Aled Edwards and Steve Muir. Experiences implementing a high-performance TCP in user-space. In *Proc. of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 196–205, September 1995.

- [20] D. Engler et al. Checking system rules using system-specific, programmer-written compiler extensions. In *Proc. of the 4th Symp. on Operating Systems Design and Implementation*, October 2000.
- [21] D. R. Engler et al. Exokernel: An operating system architecture for application-level resource management. In *Proc. of the 15th ACM Symp. on Operating Systems Principles*, pages 251–266, December 1995.
- [22] Dawson R. Engler and M. Frans Kaashoek. DPF: Fast, flexible message demultiplexing using dynamic code generation. In *Proc. of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 53–59, August 1996.
- [23] R. Greenblatt. The LISP machine. Technical Report Working Paper 79, MIT Artificial Intelligence Laboratory, November 1974.
- [24] Dan Grossman and Gregg Morrisett. Scalable certification for typed assembly language. In *Proc. of the 2000 ACM SIGPLAN Workshop on Types in Compilation*, September 2000.
- [25] Linley Gwennap. Cisco rolls its own NPU, November 2000. Microprocessor Report 11/6/00-02.
- [26] G. Hamilton and P. Kougiouris. The Spring nucleus: A microkernel for objects. In *Proc. of the Summer USENIX Technical Conference*, pages 147–160, June 1993.
- [27] Richard A. Hammond. Experiences with the Series/1 Distributed System. In *Proc. of the 21st IEEE Computer Society Int’l Conference (COMPCON 80)*, pages 585–589, Fall 1980.
- [28] S. M. Hand. Self-paging in the Nemesis operating system. In *Proc. of the 3rd Symp. on Operating Systems Design and Implementation*, pages 73–86, February 1999.
- [29] H. Hartig et al. The performance of  $\mu$ -kernel based systems. In *Proc. of the 16th ACM Symp. on Operating Systems Principles*, pages 66–77, December 1997.

- [30] K. Harty and D. R. Cheriton. Application-controlled physical memory using external page-cache management. In *Proc. of the 5th Int'l. Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 187–199, October 1992.
- [31] John L. Henning. SPEC CPU2000: Measuring CPU performance in the new millennium. *IEEE Computer*, 33(7):28–35, July 2000.
- [32] Luke Hornof. Self-specializing mobile code for adaptive network services. In *Proc. of the Int'l Working Conference on Active Networks*, volume 1942 of *Lecture Notes in Computer Science*. Springer-Verlag, October 2000.
- [33] Luke Hornof and Trevor Jim. Certifying compilation and run-time code generation. *Journal of Higher-Order and Symbolic Computation*, 12(4), December 1999.
- [34] N. C. Hutchinson and L. L. Peterson. The *x*-kernel: An architecture for implementing network protocols. *IEEE Transactions on Software Engineering*, 17(1):64–76, January 1991.
- [35] IBM Corporation. *Enterprise Systems Architecture/390: Principles of Operation*. IBM Corporation, 1997. Document number SA22-7201-04.
- [36] Intel Corporation. Intel IXP1200 network processor: Intelligent network solutions at the speed of internet growth, 2000. [download.intel.com/design/network/prodbrf/279014.pdf](http://download.intel.com/design/network/prodbrf/279014.pdf).
- [37] Internet Software Consortium. BIND (Berkeley Internet Name Domain). [www.isc.org/products/BIND/](http://www.isc.org/products/BIND/).
- [38] V. Jacobson. Compressing TCP/IP headers for low-speed serial links, February 1990. RFC-1144.
- [39] M. F. Kaashoek et al. Application performance and flexibility on Exokernel systems. In *Proc. of the 16th ACM Symp. on Operating Systems Principles*, pages 52–65, October 1997.

- [40] Robert E. Kahn and Vinton G. Cerf. What is the Internet (and what makes it work), December 1999. [www.worldcom.com/about\\_the\\_company/cerfs\\_up/internet\\_history/whatIs.phtml](http://www.worldcom.com/about_the_company/cerfs_up/internet_history/whatIs.phtml).
- [41] Eddie Kohler et al. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.
- [42] Ian Leslie et al. The design and implementation of an operating system to support distributed multimedia applications. *IEEE/ACM Journal on Selected Areas in Communications*, 14(7):1280–1297, September 1996.
- [43] J. Liedtke. Improving IPC by kernel design. In *Proc. of the 14th ACM Symp. on Operating Systems Principles*, pages 175–188, December 1993.
- [44] Barbara Liskov et al. *CLU Reference Manual*. Springer-Verlag, New York, NY, 1981.
- [45] H. Lycklama and D. L. Bayer. The MERT operating system. *The Bell System Technical Journal*, 57(6):2049–2086, July/August 1978.
- [46] Stephen Manley and Margo I. Seltzer. Web facts and fantasy. In *USENIX Symposium on Internet Technologies and Systems*, December 1997.
- [47] Henry Massalin. *An Efficient Implementation of Fundamental Operating System Services*. PhD thesis, Columbia University, 1992.
- [48] Steven McCanne and Van Jacobson. The BSD packet filter: A new architecture for user-level packet capture. In *Proc. of the Winter USENIX Conference*, pages 259–269, January 1993.
- [49] Larry McVoy and Carl Staelin. lmbench: Portable tools for performance analysis. In *Proc. of the Annual USENIX Technical Conference*, pages 279–294, January 1996.
- [50] George H. Mealy. Operating systems. Technical report, Rand Corporation, May 1962.
- [51] Paul Menage. Rcane: A resource controlled framework for active network services. In *Proc. of the 1st Int’l Working Conference on Active Networks*, June 1999.

- [52] C. W. Mercer, S. Savage, and H. Tokuda. Processor capacity reserves: An abstraction for managing processor usage. In *Proc. of the 4th Workshop on Workstation Operating Systems (WWOS-IV)*, pages 129–134, October 1993.
- [53] M. M. Michael and M. L. Scott. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In *Proc. of the 15th Annual Symposium on Principles of Distributed Computing*, pages 267–275, May 1996.
- [54] Geoffrey Milord. Segment protection with the Piglet OS. Master’s thesis, University of Pennsylvania, December 2000.
- [55] Mindcraft, Inc. WebStone: The benchmark for web servers, 2000. [www.mindcraft.com/webstone/](http://www.mindcraft.com/webstone/).
- [56] P. Mockapetris. Domain names—concepts and facilities, November 1983. RFC 882.
- [57] P. Mockapetris. Domain names—implementation and specification, November 1983. RFC 883.
- [58] J. C. Mogul and A. Borg. The effect of context switches on cache performance. In *Proc. of the 4th Int’l. Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 75–84, April 1991.
- [59] J. C. Mogul and K. K. Ramakrishnan. Eliminating receive livelock in an interrupt-driven kernel. *ACM Transactions on Computer Systems*, 15(3):217–252, August 1997.
- [60] Greg Morrisett et al. From System F to typed assembly language. *ACM Transactions on Programming Languages*, 21(3):528–569, May 1999.
- [61] D. Mosberger. *Scout: A path-based operating system*. PhD thesis, University of Arizona, 1997.
- [62] D. Mosberger and L. L. Peterson. Making paths explicit in the Scout OS. In *Proc. of the 2nd Symp. on Operating Systems Design and Implementation*, pages 153–168, 1996.

- [63] Steve Muir and Jonathan Smith. AsyMOS—an asymmetric multiprocessor operating system. In *Proc. of the 1st IEEE Conference on Open Architectures and Network Programming*, June 1998.
- [64] Steve Muir and Jonathan Smith. Supporting continuous media in the Piglet OS. In *Proc. of the 8th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video*, July 1998.
- [65] Erich M. Nahum et al. Performance issues in parallelized network protocols. In *Proc. of the 1st Symp. on Operating Systems Design and Implementation*, pages 125–137, November 1994.
- [66] Greg Nelson. *Systems Programming with Modula-3*. Prentice-Hall, April 1991.
- [67] Douglas Niehaus et al. Architecture and OS support for predictable real-time systems. Technical report, Department of Computer Science, University of Massachusetts, March 1992.
- [68] E. I. Organick. *The Multics System*. MIT Press, Cambridge, MA, 1972.
- [69] V. S. Pai, P. Druschel, and W. Zwaenepoel. Flash: An efficient and portable web server. In *Proc. of the Annual USENIX Technical Conference*, June 1999.
- [70] R. Hugo Patterson et al. Informed prefetching and caching. In *Proc. of the 15th ACM Symp. on Operating Systems Principles*, pages 79–95, December 1995.
- [71] Jon Postel. Internet protocol, September 1981. RFC 791.
- [72] Jonathan B. Postel. Simple Mail Transfer Protocol, August 1982. RFC 821.
- [73] Ian Pratt and Keir Fraser. Arsenic: A user-accessible gigabit ethernet interface. [www.cl.cam.ac.uk/users/iap10/gige.ps](http://www.cl.cam.ac.uk/users/iap10/gige.ps), 2000.
- [74] K. K. Ramakrishnan. Performance considerations in designing network interfaces. *IEEE/ACM Journal on Selected Areas in Communications (Special Issue on High Speed Computer/Network Interfaces)*, 11(2), February 1993.

- [75] K. K. Ramakrishnan et al. Operating system support for a video-on-demand file service. *ACM/Springer-Verlag Journal on Multimedia Systems*, 3, March 1995.
- [76] R. Rashid et al. Mach: A foundation for open systems. In *Proc. of the 2nd Workshop on Workstation Operating Systems (WWOS-II)*, pages 109–113, September 1989.
- [77] R. Rashid et al. Mach: A system software kernel. In *Proc. of the 34th IEEE Computer Society Int'l Conference (COMPCON 89)*, pages 176–178, February 1989.
- [78] Hans Reiser. ReiserFS, January 2001. [www.reiserfs.org](http://www.reiserfs.org).
- [79] Sendmail Consortium. Sendmail. [www.sendmail.org/](http://www.sendmail.org/).
- [80] J. S. Shapiro et al. Eros: a fast capability system. In *Proc. of the 17th ACM Symp. on Operating Systems Principles*, pages 170–185, December 1999.
- [81] W. David Sincoskie and David J. Farber. The Series/1 Distributed System: Description and comments. In *Proc. of the 21st IEEE Computer Society Int'l Conference (COMPCON 80)*, pages 579–584, Fall 1980.
- [82] Jonathan M. Smith and C. Brendan S. Traw. Giving applications access to Gb/s networking. *IEEE Network*, 7(4):44–52, July 1993.
- [83] Oliver Spatscheck et al. Optimizing TCP forwarder performance. *IEEE/ACM Transactions on Networking*, 8(3):146–157, June 2000.
- [84] Daniel Stenberg et al. cURL: a client that groks the URLs, March 2001. [curl.haxx.se](http://curl.haxx.se).
- [85] M. Stonebraker. Operating system support for database management. *Communications of the ACM*, 24(7):412–417, July 1981.
- [86] Andrew S. Tanenbaum et al. Experiences with the Amoeba distributed operating system. *Communications of the ACM*, 33(12):46–63, December 1990.
- [87] C. A. Thekkath et al. Implementing network protocols at user level. In *Proc. of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 64–73, September 1993.

- [88] C. Brendan S. Traw and Jonathan M. Smith. Hardware/software organisation of a high-performance ATM host interface. *IEEE/ACM Journal on Selected Areas in Communications (Special Issue on High Speed Computer/Network Interfaces)*, 11(2):240–253, February 1993.
- [89] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. In *Proc. of the 19th Int'l Symposium on Computer Architecture*, pages 256–266, May 1992.
- [90] T. von Eicken et al. U-Net: A user-level network interface for parallel and distributed computing. In *Proc. of the 15th ACM Symp. on Operating Systems Principles*, pages 40–53, December 1995.
- [91] Curtis Yarvin, Richard Bukowski, and Thomas Anderson. Anonymous RPC: Low-latency protection in a 64-bit address space. In *Proc. of the Summer USENIX Technical Conference*, pages 175–186, June 1993.
- [92] L. Zhang. Virtual Clock: A new traffic control algorithm for packet switching networks. In *Proc. of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 19–29, September 1990.



---

# Optical Networking and Real-Time Provisioning: An Integrated Vision for the Next-Generation Internet

**Chadi Assi, Abdallah Shami, and Mohamed A. Ali,**  
**City College of the City University of New York**  
**Russ Kurtz, REALTECH Systems Corporation**  
**Dan Guo, Sorrento Networks**

---

## Abstract

This article considers the problem of real-time provisioning of optical channels in hybrid IP-centric DWDM-based networks. First, we present an overview of the emerging architectural alternatives for IP over optical networks, namely, the overlay, the peer, and the augmented models. Then lightpath provisioning issues are detailed for route selection, with a particular focus on the "routing and wavelength assignment" (RWA) problem. In particular, a broad overview is presented, with methodologies and associated algorithms for dynamic lightpath computation being outlined. Additionally, two broad constraint-based RWA algorithms for dynamic provisioning of the optical channels are presented and evaluated. Finally, the implications of implementing the proposed RWA schemes for the lightpath provisioning aspects for each of the three emerging IP-over-optical network interconnection models are examined.

---

**R**ecently, there has been a dramatic increase in data traffic, driven primarily by the explosive growth of the Internet as well as the proliferation of virtual private networks (VPNs). At the same time, the rise of optical networking, first with wavelength-division multiplexing (WDM) transmission technology and more recently with optical multiplexers and optical cross-connect (OXC) devices, is moving us toward the vision of creating an "all-optical" Internet. In particular, these technologies yield the ability to add, drop, and in effect construct wavelength-routed networks, heralding a new era in which bandwidth is relatively abundant and inexpensive. To some, a key realization of this vision will occur when lightpaths (wavelengths) can be provisioned automatically to create bandwidth between end users, with timescales on the order of minutes or seconds. Dynamic wavelength provisioning is the main focus of this article, and will help open up a whole new world of responsive, customer-driven bandwidth services [1].

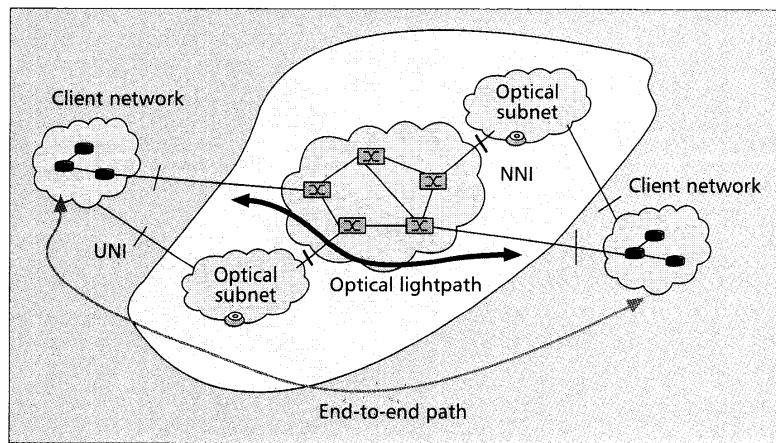
To better understand and appreciate the provisioning issue, we need to look into how circuits are currently provisioned in a typical network. Provisioning a cross-country SONET service today requires several steps. First, connectivity from the customer premise to the carrier's POP must be established for each end of the circuit. Second, a physical path must be

mapped out between the many physical hubs in a carrier's network between the two points. Each path must be checked for fiber/ring bandwidth availability. Terminating equipment must be ordered and installed on each end of each fiber path, and each interconnect point must have capacity on the optical cross-connect system. All of the cross-connects and physical interconnects must then be made and each segment documented and tested. This process is extremely manual and generally takes several months to accomplish. DWDM complicates this process even further because tens and soon hundreds of wavelengths are supported on individual fiber strands. Clearly, an automated optical routing layer will facilitate much faster provisioning.

Before this vision can be realized, however, networks need to slim down. Today's core network architecture model has four layers: IP and other content-bearing traffic; over ATM for traffic-engineering; over SONET for transport; and over WDM for fiber capacity. This approach has significant functional overlap among its layers and typically suffers from the lowest common denominator effect where any one layer can limit the scalability of the entire network. When first conceived, this layering made sense, but as IP and DWDM evolve, a more efficient interworking is called for, i.e., one that exploits the complementary features of each domain. In effect, high-performance routers plus a smart optical transport layer equipped with a new breed of photonic networking components and subsystems together are setting the foundation for the next-generation networking paradigm.

---

*This work was supported in part by DARPA/Air Force as a part of the NGI initiative under cooperative agreement number F30602-00-2-0501.*



■ Figure 1. Optical networking model.

The solution, many believe, is to layer IP directly over the optical substrate [2]. If IP can be mapped directly onto the WDM layer, some of the functional overlap can be eliminated, potentially collapsing today's vertically layered network architecture into a horizontal model where all network elements work as peers to dynamically establish optical paths. To bring the IP and WDM layers together, however, new capabilities must be added to both layers. A framing standard is needed for carrying packets directly over lambdas. Signaling standards are needed so that IP devices can control optical resources [3]. More importantly, with the conventional multi-layered architecture out of the way, automated provisioning systems will gain direct access to WDM resources, and dynamic lightpath provisioning will become easier and more practical to implement.

Once the view about network topology has changed, one will have to re-think routing as well. For example, initially there was fixed routing over fixed circuits (PSTN), and next came dynamic routing over fixed circuits (IP). Subsequently there was a move toward dynamic routing over virtual circuits (i.e., IP over ATM). Now, with recent advances in multi-protocol label switching (MPLS), we have label swapping over virtual circuits [4]. Furthermore, industry organizations such as the Optical Internetworking Forum (OIF) and the Internet Engineering Task Force (IETF) are now extending the MPLS-framework (Generalized-MPLS, also referred to as multi-protocol lambda switching (MPλS)) to support not only devices that perform packet switching (routers), but also those that perform switching in time (SONET), wavelength (OXCs), and space. Therefore most likely the next evolution will be label swapping over dynamic circuits or lightpaths (see [3, 5–11]).

The Internet Drafts cited above describe several dynamic routing possibilities [7–11]. The simplest is to treat the optical layer as completely separate from the IP layer. In this "overlay" model, optical transport offers only higher capacity and higher reliability. A more ambitious "integrated" model links the routing decisions at the IP layer with the dynamic reconfiguration capabilities of optical cross-connects (MPλS) [8]. The main goal of these initiatives is to provide a framework for real-time provisioning of optical channels, through combining recent advances in MPLS traffic-engineering control planes with emerging optical switching technologies in a hybrid IP-centric optical network [2].

This article considers the problem of real-time provisioning of optical channels in a hybrid IP-centric DWDM-based networking model. Provisioning in this work implies that an optical channel is successfully routed if both an active path

(working) and another alternate node/link-disjoint path (backup) are set up at the same time. Provisioning of connections requires algorithms for route selection, and signaling mechanisms to request and establish connectivity within the network along a chosen route. In particular, the problem of route selection in such wavelength-routed networks is referred to as the "routing and wavelength assignment (RWA) problem" [13]. Here, we present a review of RWA schemes and also compare the performance of two different constraint-based routing/RWA algorithms for dynamic provisioning of the optical channels. Specifically, the RWA schemes are used to compute end-to-end dedicated and shared backup paths to protect against single link/node failures. These algorithms are

examples of approaches that might be used to simplify the complex problem of dynamic lightpath computation. Methodologies and associated algorithms for dynamic lightpath computation are outlined. We present an overview of the emerging architectural alternatives of the two-layer model, referred to in the literature as "the interconnection models," for IP-over-optical networks, namely, the overlay, the peer, and the augmented models [8]. Finally, we examine the implications of implementing the proposed RWA schemes for the lightpath provisioning aspects for each of the three emerging interconnection models.

The remainder of this article is organized as follows. We present an overview of the emerging architectural alternatives of the two-layer model. The proposed dynamic RWA algorithm is also presented. We present an overview of fault-tolerant routing, and we present the simulation results. The implementation of real-time provisioning at the optical layer is then presented. Finally we offer a summary and a conclusion.

### *IP-over-optical Network Architectural Alternatives (Two-Layer Model)*

In the network model considered here, clients (e.g., IP/MPLS routers) are attached to an optical core network, and connected to their peers over dynamically switched optical paths (lightpaths) spanning potentially multiple OXCs. The interaction between the client and the optical core is over a well-defined signaling and routing interface, referred to as the user-network interface (UNI). Meanwhile, the optical core network consists of multiple OXCs interconnected by optical links in a general mesh topology. This network may be multi-vendor, where individual vendor OXCs constitute sub-networks. Each sub-network itself is assumed to be mesh-connected. The interaction between the sub-networks is over a well-defined signaling and routing interface, referred to as the network-network interface (NNI) (see Fig. 1).

Each OXC is assumed to be capable of switching a data stream from a given input port to a given output port. This switching function is controlled by appropriately configuring a cross-connect table. A lightpath is a fixed bandwidth connection between two network elements such as IP/MPLS routers established via the OXCs. Two IP/MPLS routers are logically connected to each other by a single-hop channel. This logical channel is the so called lightpath. A continuous lightpath is a path that uses the same wavelength on all links along the entire route from source to destination.

## Interconnection Models

One approach for transporting IP traffic over WDM networks is to use a multi-layered architecture comprised of an IP/MPLS layer over ATM over SONET over WDM. If an appropriate interface is designed to provide access to the optical network, multiple higher-layer protocols can request lightpaths to peers connected across the optical network. This architecture has four management layers. Another approach is to use a packet over SONET approach, doing away with the ATM layer, by putting IP/PPP/HDLC into SONET framing. This architecture has three management layers. The fact that both approaches support multiple protocols increases complexity for IP-WDM integration because of various edge interworkings required to route, map, and protect client signals across WDM sub-networks [7].

The two-layer model, which aims at a tighter integration between IP and optical layers, offers a series of advantages over the current multi-layer architecture model. MPLS [4] and its extension, G-MPLS [9], have been proposed as the integrating structure between IP and optical layers. Nevertheless, routing in non-optical and optical parts of hybrid IP networks needs to be coordinated. To examine the architectural alternatives for the two-layer model (IP-over-optical network), it is important to distinguish between the data plane and control planes over the user-network interface (UNI). The IP-over-optical network architecture is classified according to the organization of the control plane, i.e., whether there is a single integrated or separate independent monolithic routing and signaling protocol spanning the IP and optical domains. Several models have been proposed, including overlay, augmented, and peer-to-peer models [8].

*The Overlay Model* — Under the overlay model, the IP domain is more or less independent of the optical domain, that is, the IP domain acts as a client to the optical domain. The IP/MPLS routing and signaling protocols are independent of the routing and signaling protocols of the optical layer. Thus the topology distribution, path computation, and signaling protocols would have to be defined for the optical domain. In this model, the client routers request high-bandwidth connections (lightpaths) from the optical network through the UNI. The client routers are provided with no knowledge of the optical network topology or resources. In this scenario, the optical network provides point-to-point connection to the IP domain. The overlay model may be statically provisioned using a network management system or may be dynamically provisioned.

*The Peer Model* — In the peer model, the two layers are collapsed into a single integrated layer managed and traffic engineered in a unified manner. In this regard, the OXCs are treated just like any other router (IP/MPLS routers and OXCs act as peers) and there is only a single instance of a routing protocol spanning an administrative domain consisting of the core optical network and the surrounding edge devices (IP/MPLS routers, ATM switches). Thus, from a routing and signaling point of view, there is no distinction between the UNI, the NNI (network-network-interface), and any other router-to-router interface. This allows the SP edge devices to have full access to the topology of the core network. A common IGP like OSPF or IS-IS may be used to exchange topology information. The assumption in this model is that all the optical switches and the routers have a common addressing scheme.

*The Augmented Model* — In the augmented model, the IP and optical domains can be functionally separated, each running

its own routing protocol, but exchanging full reachability information across the UNI using a standard protocol. For example, IP addresses could be assigned to optical network elements and carried by optical routing protocols to allow reachability information to be shared with the IP domain to support some degree of automated discovery. This model combines the best of the peer and overlay interconnection models; it is relatively easy to deploy compared to the peer model in the near term. Also, this is a convenient solution, since it allows implementation of both provisioning and restoration procedures for optical sub-networks independent of the client network routing. In addition, this approach supports the common scenario in which the optical network and client networks are administered by different entities.

The central issue in this model is how the routing information is exchanged at the IP-optical UNI. There are two possibilities for this. The first is to consider the interdomain IP routing protocol, BGP, which may be adopted for exchanging routing information between IP and optical domains. The second is to consider the use OSPF areas (OSPF supports a two-level hierarchical routing scheme through the use of OSPF areas) to exchange routing information across the two domains [13]. On the other hand, running a protocol like BGP across the UNI may be considered too involved, at least for initial implementations of the UNI. A simpler approach would be to limit the reachability information passed through the optical network [13].

### Dynamic RWA

Provisioning of connections requires algorithms for route selection, and signaling mechanisms to request and establish connectivity within the network along a chosen route. The problem of route selection in such wavelength-routed networks is referred to as the “routing and wavelength assignment (RWA) problem,” and consists of two sub-problems. The first is the routing problem, which determines the path along which the connection can be established. The second problem is to assign a wavelength (or a set of wavelengths) on each link along the selected path (wavelength assignment problem). Real-time provisioning implies that both the path and wavelength should be chosen/assigned dynamically (dynamic RWA), depending on the network state. In general, all networking models described above, regardless, require route/wavelength computation/assignment to provision a lightpath, i.e., dynamic RWA engine.

### Overview of the RWA Problem

Given a set of connections, the problem of setting up lightpaths by routing and assigning a wavelength to each connection is called the routing and wavelength assignment (RWA) problem [12]. Typically, connection requests may be of three types: static, incremental, and dynamic [14]. With static traffic, the entire set of connections is known in advance, and the problem is then to set up lightpaths in a global fashion while minimizing network resources such as the number of wavelengths or the number of fibers in the network. Here, the RWA problem for static traffic is known as static lightpath establishment (SLE) and can be formulated as a mixed-integer linear program [15]. In the incremental-traffic case, connection requests arrive sequentially, a lightpath is established for each connection, and the lightpath remains in the network indefinitely.

For the case of dynamic traffic, a lightpath is set up for each connection request as it arrives, and the lightpath is released after some finite amount of time. The objective in the incremental and dynamic traffic cases is to set up lightpaths and assign wavelengths in a manner that minimizes the

amount of connection blocking [14]. This problem is referred to as the dynamic lightpath establishment (DLE). Generally, the DLE is more difficult to solve, and therefore heuristics methods are generally employed. Heuristics exist for both the routing sub-problem and the wavelength assignment sub-problem.

For the routing sub-problem, there are three basic approaches that can be found in the literature: fixed routing, fixed-alternate routing, and adaptive routing [14]. Fixed routing is one variant of "static routing" in which routing decisions do not vary with time. Moreover in fixed routing the same fixed route for a given source-destination pair is always selected. The fixed alternate routing approach considers multiple routes between a source-destination pair, and each node in the network maintains an ordered list of a number of fixed routes to each destination node. When a connection request arrives, the source node attempts to establish the connection on each of the routes from the list in sequence, until a route with a valid wavelength assignment is found. Conversely, in adaptive routing [14, 16–18] the route from a source node to a destination node is chosen dynamically, depending on the network state. Adaptive routing requires extensive support from the control and management protocols to continuously update the routing table at the node. An advantage of adaptive routing is that it results in lower connection blocking than fixed and fixed-alternate routing.

Meanwhile for the wavelength assignment sub-problem, a number of heuristics have been proposed [19–21]. These heuristics are Random Wavelength Assignment, First-Fit, Least-Used, Most-Used, Min-Product, Least-Loaded, MAX-SUM, Relative Capacity Loss, Wavelength Reservation, and Protecting Threshold. In [17] the authors propose an adaptive unconstrained routing (AUR), which incorporates network state information into route computation and channel allocation.

Currently, the algorithms that offer the best performance are Relative Capacity Loss (RCL) [21], and Distributed Relative Capacity Loss (DRCL) [14]. RCL calculates the Relative Capacity Loss for each path on each available wavelength and then chooses the wavelength that minimizes the sum of the relative capacity loss on all the paths. DRCL is proposed in [14] and is based on RCL but it is more efficient in a distributed environment. For a tutorial review of the RWA problem, we refer the reader to [14].

Optical networks can also pose added wavelength continuity constraints [13], and these may require the use of wavelength conversion (also referred to as wavelength translation or wavelength changing). A wavelength converter is a device that takes at its input a data channel modulated onto an optical carrier with a wavelength  $\lambda_{in}$ , and produces at its output the same data channel modulated onto an optical carrier with a different wavelength  $\lambda_{out}$ . If wavelength converters are included in the OXCs in WDM networks, connections can be established without the need to find an unoccupied wavelength, which is the same on all the links making the route. This means that networks with wavelength converters are equivalent to traditional circuit-switched networks. Wavelength converters thus result in improvements in network performance. On the other hand, it has been shown that a careful wavelength assignment in wavelength-continuous networks can lead to improved performance, thus reducing the benefits of wavelength converters [22]. In [22] the authors investigate the benefits of limited wavelength conversion for ring and mesh-torus topologies with fixed shortest path routing. The authors of [23] used a hypercube network to study limited conversion with fixed shortest path routing and a First-Fit wavelength selection algorithm. It is shown that limited wave-

length conversion (25 percent) achieves the same performance improvement as full wavelength conversion [22, 23].

Additionally, many other constraints can also serve to complicate the RWA process, especially in all-optical networks. Specifically, besides wavelength continuity requirements, these include analog attenuation effects and power limitations. For example, adequate signal-to-noise ratios (SNRs), crosstalk, and dispersion effects caused by subsystem components and fiber links can be computed along candidate paths. This information can be incorporated into route resolution strategies by defining new cost functions [24]. In [24] the authors have extended the routing and wavelength assignment problem to account for the power degradation of a routed signal due to non-ideal behavior of optical components such as multiplexers, demultiplexers, taps, and fiber links.

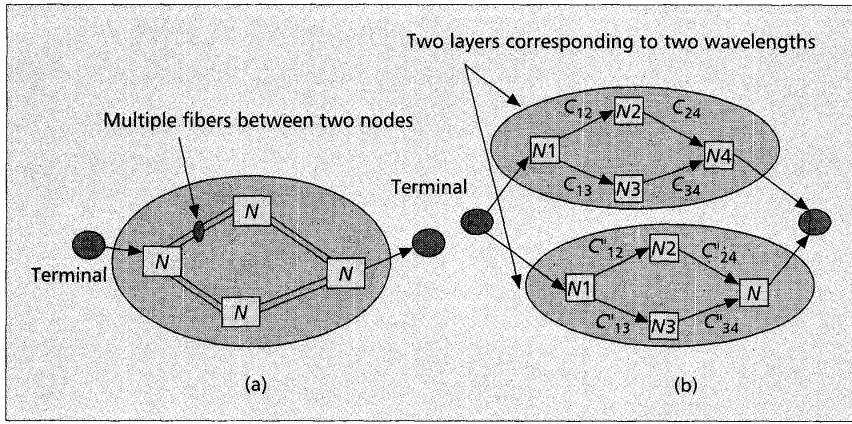
### *The Proposed Dynamic RWA Algorithms*

Some combined RWA algorithms are now presented. Specifically, these algorithms integrate and collapse both the routing and wavelength assignment sub-problems into a single dynamic constraint-based routing problem. Thus, the emphasis here is on the adaptive routing problem, rather than focusing on the wavelength-assignment problem. It has been shown that the routing scheme has much more of an impact on the overall network performance than the wavelength-assignment scheme [14, 16]. Moreover, both algorithms are also shown to be capable of supporting fault-tolerant adaptive routing and are amenable to fully distributed implementations.

The network is viewed as multi-layered graphs, each corresponding to a specific wavelength. For a connection request and on a given wavelength, Dijkstra's shortest path algorithm, which is suitably modified for WDM networks, is used for computing a constraint path. This is achieved by associating each link in the network with a specific weight function that incorporates WDM-specific information such as the number of available wavelengths and the total wavelengths [18]. This means that the algorithm might compute (on-line)  $W$  Paths, each corresponding to one of the  $W$  wavelengths. Then one of these paths is selected according to a global selection criterion. Thus the problem of wavelength assignment is totally mitigated and both the routing and wavelength assignment sub-problems are now integrated and collapsed into a single dynamic constraint-based routing problem. This is in contrast to the work reported in [18], in which a single path is first calculated, then a wavelength is assigned to the path by propagating a wavelength request to all the routers along the path. In this way such an algorithm avoids the overhead associated with such a wavelength request (probe message). The emphasis here is on the adaptive routing problem, rather than focusing on the wavelength assignment problem. It has been shown that the routing scheme has much more of an impact on the overall network performance than the wavelength assignment scheme [14–16].

The algorithm is implemented as per the following:

1. First consider a multi-fiber IP-centric DWDM-based network whose physical topology consists of multiple OXCs interconnected via point-to-point WDM links in an arbitrary mesh topology.
2. Assume that none of the OXCs has wavelength conversion capability. Hence, to meet a connection request, a lightpath, which uses the same wavelength on all the links along the entire route from source to destination, has to be set up.
3. Both algorithms are based on a fully distributed implementation in which all nodes maintain a synchronized and identical topology and link state information (traffic engineering database, TED).
4. Assuming that  $W$  is the number of wavelengths per fiber,



■ Figure 2. The sample network model (a) and its multilayered graph approach (b).

the network is represented by  $W$  identical graphs, each conforming to the physical topology and a particular wavelength. Hence the network can be viewed as  $W$  identical wavelength graphs, each representing a wavelength. In view of this multi-graph model, each physical link is now represented by  $W$  virtual links (channels), each corresponding to one of the wavelength graphs. Figure 2 illustrates the concept of the multi-graph approach for a simple network with four nodes, four physical links, and  $W = 2$ .

5. For a given connection request, a constraint route is calculated, for each of the wavelength graphs, throughout the entire network from source to destination, typically using a shortest path algorithm but with the link weights adjusted to attain some sort of local resource optimization. Clearly there are at most  $W$  paths that can be calculated, each corresponding to a given wavelength, provided that each path can meet the given routing constraint. As a result we get the vector  $V = \langle \text{Path}_i, \text{Wavelength}_i \rangle, i = 1, \dots, W$ , where the number of entries stored in  $V$  may vary from no entries at all (request is blocked) to a possible maximum of  $W$  entries. Finally, to globally optimize the network resources, provided that the number of entries stored in  $V$  is more than one, an entry ( $\text{Path}_i, \text{Wavelength}_i$ ) has to be selected out of all the other possible entries. Thus by virtually separating wavelengths, both the routing and wavelength assignment sub-problems are now reduced into a single dynamic constraint-based routing problem.

**Algorithm 1: Full Adaptive Routing** — The implementation of this algorithm is as follows:

1. For a given wavelength graph  $\lambda_i$ , each virtual link, in each of the  $W$  wavelength graphs, is assigned a cost. Basically, the cost of a link at a given wavelength  $\lambda_i$  is defined here as the inverse of the number of available channels over that particular link. Hence initially the cost of a given link throughout the entire network is set =  $(1/F)$ , where  $F$  is the number of fibers (per link) connecting two adjacent OXCs. In general, the cost of link  $L_j$  at wavelength  $\lambda_i$ ,  $C(L_j^{\lambda_i})$  is given by:

$$C(L_j^{\lambda_i}) = \begin{cases} \frac{1}{F - N(L_j^{\lambda_i})} & \text{if } N(L_j^{\lambda_i}) < F \\ \infty & \text{if } N(L_j^{\lambda_i}) = F \end{cases} \quad (1)$$

Where  $N(L_j^{\lambda_i})$  is the number of occupied (unavailable)  $\lambda_i$ 's on link  $L_j$ .

2. For a given wavelength  $\lambda_i$ , we associate each path throughout the entire network with a total cost,  $C_{sd}^{\lambda_i}$ , which is defined here as the summation of the costs of all individual links spanning the entire path from source to destination.

$$C_{sd}^{\lambda_i} = \sum_{j=1}^n C(L_j^{\lambda_i}) \quad (2)$$

where  $\langle L_1, L_2, \dots, L_n \rangle$  is the set of  $n$  links that comprise the path.

3. For a given connection request, run Dijkstra's algorithm on the first wavelength graph  $\lambda_1$  to find the shortest path (the path with minimum  $C_{sd}^{\lambda_i}$ ). Store the calculated path along with its corresponding wavelength  $\lambda_1$  as the first entry of the vector  $V$ . Note that the calculated local path, for a given wavelength graph  $\lambda_i$ , is not necessarily the path with the mini-

mum number of hops.

4. Repeat Step 3 for each of the remaining  $W-1$  wavelength graphs. Note that the vector  $V$  might now have up to  $W$  entries.

5. Examine the contents of the vector  $V$  and perform one of the following instructions:

- If the vector  $V$  has no entries at all, reject the connection request; otherwise go to step # b.
- If the vector  $V$  has only one entry, select this entry as the combination ( $\text{Path}_i, \text{Wavelength}_i$ ) that satisfies the connection request. After assigning the path, update the weights associated with all links along the entire path (only on the corresponding wavelength graph  $\lambda_i$ ) by basically decrementing the number of the available  $\lambda_i$ 's (channels) on every link along the selected path by one; otherwise go to step # c.
- If the vector  $V$  has more than one entry, select one of those entry combinations that satisfy one of the following global path selection schemes:

**Total Cost-Based Selection** — In this scheme a total cost,  $C_{sd}^{\lambda_i}$ , is associated with each computed path within the vector  $V$  ( $\langle \text{Path}_i, \text{Wavelength}_i \rangle$ ), given by:

$$C_{sd}^{\lambda_i} = \sum_{j=1}^n C(L_j^{\lambda_i}) \quad (3)$$

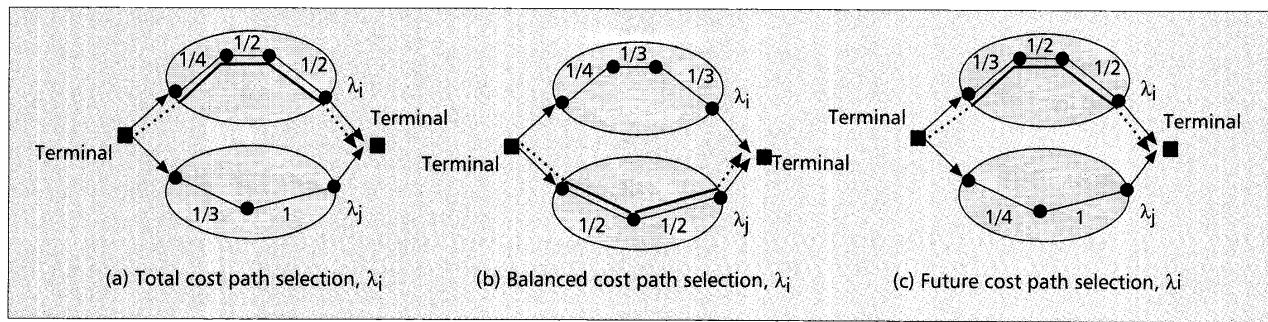
The path with the minimum total cost  $C_{sd}^{\lambda_i}$  is selected and assigned to the connection. Note that this selection criterion skews the conventional shortest path search to favor less utilized network resources. This is illustrated in Fig. 3a, where the path with fewer hops is not selected.

**Balanced Cost-Based Selection** — In this case each path is assigned a balanced cost,  $C_{sd}^{B\lambda_i}$ , defined by:

$$C_{sd}^{B\lambda_i} = n \times \left( \sum_{j=1}^n C(L_j^{\lambda_i}) \right)$$

where  $n$  is the total number of links along the path.

The path with the least balanced cost  $C_{sd}^{B\lambda_i}$  is selected and assigned to the connection. Note that this selection criterion strikes a balance between the minimum cost and the minimum number of hops. The main objective of this selection criterion is to avoid assigning long paths to a connection and to route connections over the healthy part of the network. This is illustrated in Fig. 3b, where the connection is routed on the path with wavelength  $\lambda_j$ , which has a balanced cost ( $2*1$ ) less than that with wavelength  $\lambda_i$  [ $3*(11/12)$ ], thus avoiding assigning the connection to the longer path ( $\lambda_i$ ). Note,



■ Figure 3. Illustration examples.

however, that the total cost of the path with wavelength  $\lambda_j$  (1) is higher than that with  $\lambda_i$  (11/12).

**Future Cost-Based Selection** — This scheme uses the same total cost  $C_{sd}^{\lambda_i}$  of Eq. (3), but with the individual link cost  $C(L_j^{\lambda_i})$  of Eq. (1) redefined as:

$$C(L_j^{\lambda_i, \text{Future}}) = \frac{1}{F - 1 - N(L_j^{\lambda_i})} \quad \text{if } N(L_j^{\lambda_i}) < |F|$$

Thus the total future cost of this scheme is given by:

$$C_{sd}^{\lambda_i, \text{Future}} = \sum_{j=1}^n C_{L_j}^{\lambda_i, \text{Future}}$$

The path with the minimum future cost is selected and assigned to the connection. If all paths have the same future cost of infinity (i.e., along a path at least one link has one last channel available), select the path with the least cost using the original definition of individual link cost of Eq. (1). Note that this selection criterion strongly favors assigning paths with the least current/future utilized resources. This is illustrated in Fig. 3c, where the path with infinity future cost is not selected.

Two comments are in order here. First, for any of the three selection schemes described above, after a path is assigned the weights associated with all links along the selected path should be updated as indicated in step 5b above. Second, all link state updates must be propagated and advertised to all other core nodes throughout the network. Note that the frequency of the link state update per unit time is proportional to the number of accepted/released calls. In other words, the link state update is triggered once for every accepted/released call. As a result, the signaling overhead associated with a high volume of calls may be excessive.

**Algorithm II: Semi-Adaptive Routing** — This algorithm adopts the same implementation procedures developed for the full adaptive algorithm described above, except for the following fundamental differences:

1. A shortest path algorithm (Dijkstra's algorithm) is initially run off-line to calculate the shortest path (only minimum number of hops) between every source-destination node pair (routing tables) throughout the entire network. These off-line computed routing tables are stored at each node in every wavelength graph. Thus the initial routing tables are identical for all  $W$  wavelength graphs.

2. For an initial connection request the ingress node at every wavelength graph consults its own routing table for the shortest path. As a result, similar to algorithm I, we may get as much as  $W$  paths, where one of them can then be selected according to the selection schemes described above.

3. For all the consecutive connection requests, the routing tables remain unchanged, so that Step 2 is repeated until the cost of a link  $L_j$  in a given wavelength graph  $\lambda_i$  goes to infinity

(no more available  $\lambda_i$ 's). In this case link  $L_j$  is removed from wavelength graph  $\lambda_i$  and the routing tables are calculated for each node again. Note that Dijkstra's algorithm is run this time online to find the shortest path (the path with minimum  $C_{sd}^{\lambda_i}$ ).

Note that the signaling overhead associated with the link state updates for this algorithm is considerably less than that of algorithm I (link state updates are only triggered when the cost of the link goes to infinity). In addition the time associated with computing a path for the semi-adaptive algorithm is less than that of the full adaptive algorithm, since the path is directly read off the routing table.

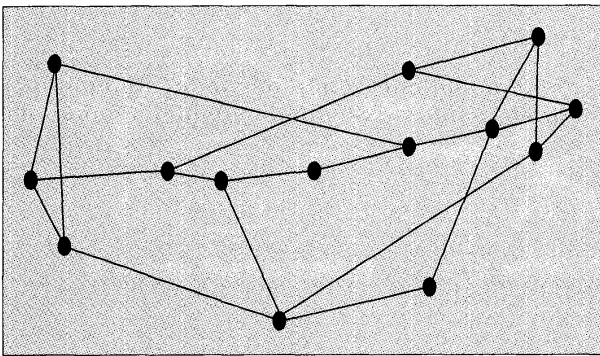
Finally, in the case where the lightpath is wavelength-continuous, as this work has assumed, optical non-linearities, chromatic dispersion, amplifier spontaneous emission, and other factors together limit the scalability of an all-optical network. Routing in such networks will then have to take into account noise accumulation and dispersion to ensure that lightpaths are established with adequate signal qualities. This work assume that the all-optical (sub-)network considered is geographically constrained so that all routes will have adequate signal quality, and physical layer attributes can be ignored during routing and wavelength assignment. However, the policies and mechanisms proposed here can be extended to account for physical layer characteristics, and requires future work.

## Fault-Tolerant Routing

Given the wide range of services envisioned for future IP networks, network survivability is a crucial concern. Survivability schemes can be classified into two forms: protection, which refers to pre-provisioned failure recovery; and restoration, which refers to more dynamic signaled recovery [2]. A common approach to protection is to set up two physically link-disjoint paths for every connection request. One path, called the primary, is used to transmit data, while the other path is reserved as a backup in the event that a link in the primary path fails. To further protect against node failures, the primary and backup paths may also be node-disjoint [25, 26].

Fixed-alternate routing provides a straightforward approach to handling protection [17]. On the other hand, in adaptive routing, a protection scheme may be implemented in which the backup path is set up immediately after the primary path has been established [14]. The same routing protocol may be used to determine the backup path, with the exception that a link cost is set to infinity if that link is being used by the primary path. The resulting route will then be link-disjoint from the primary path. Since these schemes require backup path routing at setup time, they must be more closely incorporated with the primary lightpath RWA algorithms. On the other hand, channel restoration does not rely on pre-computed backup routes, and instead dynamically re-computes a new path for a broken channel [26]. This has the advantage of low overhead in the absence of failures. However; this does not guarantee successful recovery, since the attempt to establish a new path may fail





■ Figure 4. 14-node NSFNET topology.

due to resource shortage at the time of failure recovery. Additionally, recovery timescales are usually longer [2].

By making use of the WDM channel routing capabilities, a variety of lightpath protection schemes can be designed. For example, dedicated backup channels can be provisioned for users requiring high availability. Here a pre-computed link-disjoint backup channel is reserved for each primary channel at setup time, and in case of a fault condition on the primary path, a channel switchover is performed. The dedicated backup reservation method has the advantage of shorter restoration time since the resources are reserved for the backup path when establishing the primary path itself. However, this method reserves excessive resources. For better resource utilization, multiplexing techniques [27] can be employed. If two primary lightpaths do not fail simultaneously, their backup lightpaths can share a wavelength channel. However, in case of primary link failure, the backup capability of the other is no longer preserved. Therefore, although channel-blocking rates will be less (than the dedicated case), channel recovery probability will also be lower.

In addition, recently the concept of a shared risk link group (SRLG) definition has also been proposed to help identify risk associations between various entities (see [28]). This concept is used to ensure that the primary and the backup path are not affected by the same failure. By using this concept, adequate resource "disjointness" can be introduced into the constraint-based path computation phase, thereby reducing the probability of simultaneous lightpath failures (e.g., between working and protection paths). Further details are beyond the scope of this article, and interested readers are referred to [8, 28].

Overall, the proposed adaptive RWA scheme can be extended to ensure diversity in routes. This can be achieved by coordinating each diversely routed lightpath group by a single network entity. To create a diversely routed lightpath group, a user registers with a coordinator and receives the group identifier. For groups originating through the same client router, this router would typically act as the coordinator. To ensure diversity in routes,  $N$  SRLG and node disjoint routes through the network are selected, where  $N$  represents the number of diverse routes required.

## Simulation Results

The performance of the proposed dynamic RWA algorithms is evaluated via simulation of the mesh-based NSFNET shown in Fig. 4. The NSFNET consists of 14 nodes and 21 physical links. Each adjacent node pair is connected through a bi-directional physical link that consists of  $N$  fibers, where each fiber is assumed to have the same number of wavelengths ( $W$ ). The simulation results calculated in this section assume that  $N = 2$  and  $W = 4$ . We use a dynamic traffic model in which call requests arrive at each node according to a Poisson process

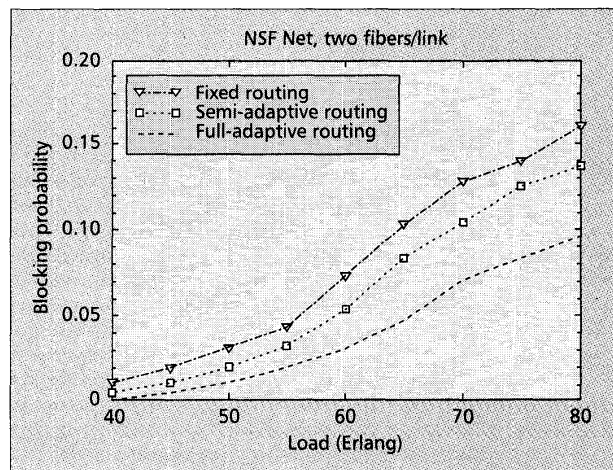
with a network arrival rate  $\lambda$ . An arrival session is equally likely to be destined to any node in the network. The session holding time is assumed to be exponentially distributed with mean  $1/\mu$ . The blocking probability is the metric used to evaluate the network performance. In each simulation run, a large number of requests are generated one after the other, and the results are averaged over many simulation runs. If at any time a connection request cannot be satisfied according to the algorithms developed above, the connection request is dropped.

Figure 5 shows the simulated blocking probability vs. the call arrival rate for both algorithms when the total cost-based path selection criterion (scheme I) is used. The simulated blocking probability is also shown in the figure for the conventional static RWA scheme used in most algorithms. As expected, it can be seen from the figure that the performance of both dynamic algorithms is significantly better than that of the static algorithm. Note, however, that the performance of the full-adaptive algorithm is slightly better than that of the semi-adaptive algorithm. These results always hold, independent of which path selection scheme described above (I, II, or III) is used.

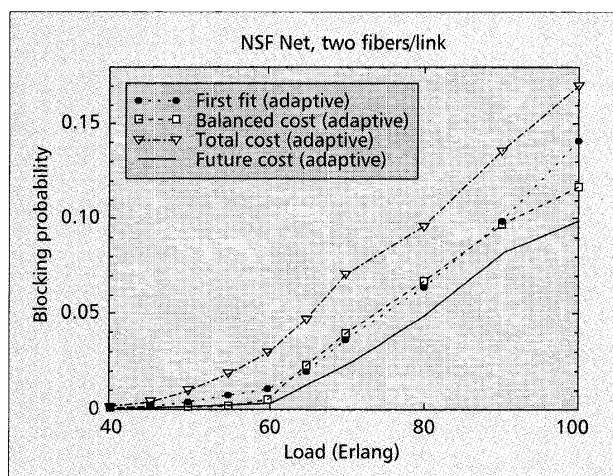
Figure 6 shows the simulated blocking probability vs. the call arrival rate of the full-adaptive algorithm for all three path selection schemes described above. As can be seen from Fig. 6, the path selection process based on future cost performs the best, followed by the path selection process based on the balanced cost, then by the path selection process based on total cost. Also included in the figure is the adaptive First Fit (FF) algorithm [14]. Note that the performance of the future cost algorithm is significantly better than that of the FF. However, the performance of the balanced cost algorithm is almost the same as that of the FF algorithm (same results were also obtained in [14] with the DRCL algorithm), and both of them outperform the total cost scheme.

Figure 7 shows the required number of wavelengths vs. the call arrival rate for three different cases:

- A system that uses end-to-end dedicated backup paths to provide 100 percent protection against single link/node failures.
- A system that uses end-to-end shared backup paths to provide 100 percent protection against single link/node failures.
- A system that provides no protection at all. As expected, the number of wavelengths required to provide shared protection is considerably less than that required for the dedicated case (also reported in [26]).



■ Figure 5. Blocking probability vs. load, for different routing schemes.



■ Figure 6. Comparison of the different selection schemes for the adaptive routing case.

## Real-time Provisioning at the Optical Layer

Provisioning end-to-end circuits is an endless source of struggle for service providers and a frustration for end-users. Provisioning of connections requires algorithms for route selection, and signaling mechanisms to request and establish connectivity within the network along a chosen route. In this section we examine the problem of route selection in the context of applying/adapting the RWA algorithm presented above to each of the three interconnection models described earlier. The implications on both the route selection and signaling mechanism components will also be outlined for each of the three interconnection models.

### Dynamic Lightpath Computation

Dynamic computation of a lightpath involves the implementation of two traffic engineering components: an information distribution mechanism that provide knowledge of the relevant attributes of available network resources, and a path selection process that uses the information distributed by the dynamic link state advertisement algorithm to select a path that meets the specific requirements of the traffic flow. In a fully distributed IP-over-optical network implementation, these are:

**An information distribution mechanism:** Provides knowledge of the network's topology and the available resources. This component is implemented by defining relatively simple extensions to the interior gateway protocol (IGP), e.g., open shortest path first (OSPF) so that link attributes are included as part of each router's link state advertisement. Some of the traffic-engineering extensions that need to be added to the IGP link state advertisement include maximum link bandwidth, maximum reservable link bandwidth, current bandwidth reservation, current bandwidth usage, and link coloring [29]. These extensions capture optical link parameters and any constraints specific to optical networks. Such topology and link state information is then flooded to all nodes via updates. Another important component is to define a naming and addressing convention for different elements of the physical plant hierarchy [11]. Here we have defined and assigned a naming and addressing convention for different elements of the physical plant hierarchy, i.e., by implementing a simplified link state advertisement algorithm to model extended OSPF. This algorithm is capable of periodically updating and advertising all of the above link attributes. The link state updates can be triggered, for instance, based on a given threshold of the number of available wavelengths per fiber, below which the updates can be triggered. Once each node has a representation of the full physical network topology

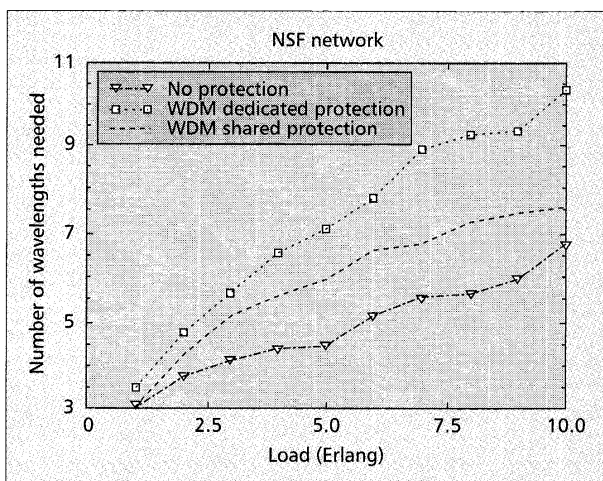
and the available resources on each link, a path selection algorithm is required, i.e., dynamic RWA.

**A path selection process:** Uses the information distributed by the dynamic link state advertisement algorithm to select an explicit route that meets the specific requirements of the traffic flow. This process can be performed either off-line or online using a constraint-based routing calculation. The source router (peer model) or the border OXC/central management node (augmented/overlay models) are basically responsible for computing the complete path all the way to the destination through the optical domain, and then initiating path setup using the signaling protocol (e.g., CR-LDP or RSVP). The route may be specified either as a series of nodes (routers/OXCs) or in terms of the specific links used (as long as IP addresses are associated with these links).

### Route Selection Using the Proposed RWA

Numerous policies can be used to route lightpaths through the network, such as the constraint-based routing algorithms proposed here. This scheme can be used directly for computing the route and assigning the wavelength for both the overlay and the augmented models. In this case a connection request is initiated by a client IP/MPLS router (border router, that is, a router directly connected to the optical network) and sent to an ingress optical node (border OXC, that is, the OXC connected to the border router) using UNI signaling. Such provisioning requests may specify the desired destination client router. Note that the source end-point is implicit in this case. The ingress optical node processes the request and computes an appropriate route along with a wavelength through the network (using topology and state information that has been propagated using OSPF link state advertisements). Note that the request may also be received by an ingress OXC from a central management node, specifying the source and destination end-points.

The routing within the optical and IP domains in the case of the augmented model may be separated, with a standard routing protocol running between domains [7, 8]. This is similar to the IP interdomain routing model, where the central issue is how the routing information is exchanged at the IP-optical UNI. There are two possibilities for this. The first is to consider the interdomain IP routing protocol, BGP, which may be adopted for exchanging routing information between IP and optical domains. The second is to consider the use of OSPF areas (OSPF supports a two-level hierarchical routing



■ Figure 7. Comparison of the dedicated protection vs. shared protection.



scheme through the use of OSPF areas) to exchange routing information across the two domains [7, 8].

However, in the case of the peer model additional extensions need to be added to the routing protocol (OSPF) so that the segment of the entire route that crosses the optical core (between the ingress and egress OXCs) must be treated as a virtual link of fixed capacity and advertised as such in further OSPF updates. The routing in this case is referred to as a "flat" routing organization [7-8]. Under this approach there is only one instance of the routing protocol running in the IP and optical domains. An IGP such as OSPF or IS-IS with suitable optical extensions is used to exchange topology information. These optical extensions will capture the unique optical link parameters. The OXCs and the routers maintain the same link state database. The routers can then compute end-to-end paths to other routers across the OXCs. This lighpath is always a tunnel across the optical network between edge routers. Once created such lighpaths are treated as virtual links and are used in traffic engineering and route computation. As and when forwarding adjacencies (FAs) are introduced in the link state, corresponding links over the IP optical interface are removed from the link state advertisements. Finally, the details of the optical network are completely replaced by the FAs advertised in the link state [7, 8].

### RWA Implications on Signaling Mechanisms

Once a lighpath request from a source is received by the ingress node, it computes the complete path all the way to the destination through the optical domain using the proposed RWA algorithm. The output of this calculation is an explicit route consisting of a sequence of hops that provides the shortest path through the network that meets the constraints. This explicit route is then passed to the signaling component that initiates path setup (to reserve resources) using the signaling protocol, e.g., CR-LDP or RSVP-TE [9]. Note that the implications of using the proposed RWA scheme on the signaling is that the conventional overhead associated with the wavelength request (probe message) is no longer needed, since the RWA scheme selects the route and assigns the wavelength simultaneously.

### Conclusion

This article has considered the problem of real-time provisioning of optical channels in a hybrid IP-centric DWDM-based networking model. Provisioning implies that an optical channel is successfully routed if both an active path (working) and another alternate link-disjoint path (backup) are set up at the same time. Specifically, the work presented here has addressed the implementation issues of the path selection component of the traffic engineering problem in such a network. Methodologies and associated algorithms for dynamic lighpath computation were outlined.

We have presented and compared the performance of two different constraint-based routing and wavelength assignment (RWA) algorithms for dynamic provisioning of the optical channels. Specifically, the RWA scheme is used to compute end-to-end dedicated and shared backup paths to protect against single link/node failures. Three path selection schemes have also been proposed for each algorithm. Both algorithms are based on a fully distributed implementation. The performance of both algorithms was then compared with that of the conventional static RWA algorithm. It was shown that the dynamic full-adaptive algorithm outperforms the semi-adaptive algorithm, and both algorithms significantly outperform the conventional static algorithm. It was also shown that the Future Cost-Based Selection scheme outperforms both the total-based and the balanced selection schemes.

### Acknowledgment

The authors would like to thank Nasir Ghani and Debanjan Saha for their helpful comments and discussions.

### References

- [1] J. M. Elmirghani and H. Mouftah, "Technologies and Architectures for Scalable Dynamic Dense WDM Networks," *IEEE Commun. Mag.*, vol. 38, no. 2, Feb. 2000, pp. 58-66.
- [2] N. Ghani *et al.*, "On IP-over-WDM Integration," *IEEE Commun. Mag.*, Mar. 2000.
- [3] N. Ghani, "Lambda-Labeling: A Framework for IP-over-WDM using MPLS," *Optical Networks Magazine*, vol. 1, Apr. 2000, pp. 45-58.
- [4] R. Callon, "A Framework for Multi-Protocol Label Switching," work in progress, Sept. 1999.
- [5] D. Awduche *et al.*, "Multiprotocol Lambda Switching, Combining MPLS Traffic Engineering with Optical Crossconnects," work in progress, Internet Draft, draft-awduche-mpls-te-optical-01.txt.
- [6] Y. Ye, S. Dixit, and M. A. Ali, "On Joint Protection/Restoration in IP-Centric DWDM-based Optical Transport Networks," *IEEE Commun. Mag.*, vol. 38, no. 6, June 2000, pp. 174-83.
- [7] N. Chandhok *et al.*, "IP-over-optical Networks: A Summary of Issues," work in progress, Internet Draft, draft-osu-ipo-mpls-issues-02.txt.
- [8] B. Rajagopalan *et al.*, "IP-over-optical Networks: A Framework," work in progress, Internet Draft, draft-many-optical-framework-02.txt.
- [9] P. Ashwood-Smith *et al.*, "Generalized MPLS, Signaling Functional Description" work in progress, Internet Draft, draft-ietf-mpls-generalized-signaling-01.txt.
- [10] B. Rajagopalan *et al.*, "IP-over-optical Networks: Architecture Aspects," *IEEE Commun. Mag.*, Sept. 2000, pp. 94-102.
- [11] S. Chaudhuri, G. Hjalmtysson, and J. Yates, "Control of Lighpaths in an Optical Network," OIF submission OIF2000.04, Jan 2000 and IETF draft, Feb. 2000.
- [12] B. Mukherjee, *Optical Communications Networks*, McGraw-Hill, NY, 1997.
- [13] D. Pendarakis, B. Rajagopalan, and D. Saha, "Routing Information Exchange in Optical Networks," work in progress, Internet Draft, draft-prs-optical-routing-01.txt.
- [14] H. Zang, J. P. Jue, and B. Mukherjee, "A Review of Routing and Wavelength Assignment Approaches for Wavelength-Routed Optical WDM Networks," *Optical Networks Magazine*, vol. 1, Jan. 2000, pp. 47-60.
- [15] R. Ramaswami and K. N. Sivarajan, "Dynamic Allocation in All-Optical Ring Networks," *Proc. IEEE ICC '97*, Montreal, Quebec, Canada, vol. 1, June 1997, pp. 432-36.
- [16] S. Ramamurthy and B. Mukherjee, "Fixed-Alternate Routing and Wavelength Assignment in Wavelength Routed Optical Networks," *Proc. IEEE GLOBECOM '98*, vol. 4, Nov. 1998, pp. 2295-302.
- [17] A. Mokhtar and M. Azizoglu, "Adaptive Wavelength Routing in All-Optical Networks," *IEEE/ACM Trans. Net.*, vol. 6, Apr. 1998, pp. 197-206.
- [18] T. Fabry-Asztalos, N. Bhide, and K. M. Sivalingam, "Adaptive Weight Functions for Shortest Path Routing Algorithms for Multi-Wavelength Optical WDM Networks," *Proc. IEEE ICC2000*, New Orleans, LA, June 2000.
- [19] I. Chlamtac, A. Ganz, and G. Karmi, "Purely Optical Networks for Terabit Communication," *Proc. IEEE INFOCOM '89*, Washington, DC, vol. 3, Apr. 1989, pp. 887-96.
- [20] E. Karasan and E. Ayanoglu, "Effects of Wavelength Routing and Selection Algorithms on Wavelength Conversion Gain in WDM Optical Networks," *IEEE/ACM Trans. Net.*, vol. 6, no. 2, Apr. 1998, pp. 186-96.
- [21] X. Zhang and C. Qiao, "Wavelength Assignment for Dynamic Traffic in Multi-Fiber WDM Networks," *Proc. 7th ICCCN*, Oct. 1998, pp. 479-85.
- [22] J. Yates, P. Rumsewicz, and J. Lacey, "Wavelength Converters in Dynamically Reconfigurable WDM Networks" *IEEE Communications Surveys*, Q2, 1999.
- [23] H. Harai, M. Murata, and H. Miyahara, "Performance of All-Optical Networks with Limited Range Wavelength Conversion," *Proc. IEEE ICC*, Montreal Canada, 1997, Apr. 1997, pp. 416-21.
- [24] M. Ali *et al.*, "Routing and Wavelength Assignment (RWA) with Power Considerations in All-Optical Wavelength-Routed Networks," *Proc. IEEE GLOBECOM '99*, Rio De Janeiro, Brazil, Dec. 1999, pp. 1437-43.
- [25] R. Bahandary, *Survivable Networks: Algorithms for Diverse Routing*, Kluwer Academic Publishers [1999].
- [26] B. Doshi *et al.*, "Optical Network Design and Restoration," *Bell Labs Tech. J.*, vol. 4, no. 1, Jan-Mar 1999, pp. 58-84.
- [27] G. Mohan and C. S. R. Murthy, "Lighpath Restoration in WDM Optical Networks," *IEEE Network*, Nov./Dec. 2000.
- [28] B. Rajagopalan and D. Saha, "Link Bundling in Optical Networks," work in progress, Internet draft, draft-rs-optical-bundling-01.txt.
- [29] K. Kompella *et al.*, "OSPF Extensions in Support of GMPLS," work in progress, Internet draft, draft-kompella-ospf-gmpls-extensions-00.txt.

### Additional Reading

- [1] N. Ghani *et al.*, "Architecture Framework for Automatic Protection Provisioning in Dynamic Optical Rings" work in progress, Internet draft, draft-ghani-optical-rings-01.txt.

## Biographies

CHADI M. ASSI (assi@ees1s0.engr.cuny.cuny.edu) received the B.E. degree in electrical and electronics engineering from the Lebanese University, Beirut, in 1997, and the M.S. degree in electrical engineering from the City University of New York in 1999. He is currently pursuing the Ph.D. degree in electrical engineering at the City University of New York. In the summer of 2000 he was an intern at Sorrento Networks, San Diego, where he developed software for the RWA and wavelength conversion in optical networks. His current research interests are in the area of optical networking, network architectures, IP over WDM, and specifically protection/restoration and real-time provisioning.

ABDALLAH A. SHAMI (shami@ees1s0.engr.cuny.cuny.edu) received the B.E. degree in electrical and electronics engineering from the Lebanese University, Beirut, in 1997, the M.S. degree in electrical engineering from the Saint Joseph University in Lebanon in 1998. He is currently pursuing the Ph.D. degree in electrical engineering at the City University of New York. In the summer of 2000 he was an intern at Sorrento Networks, San Diego, where he developed software for the RWA and wavelength conversion in optical networks. His current research interests are in the area of optical networking, IP over WDM, and specifically traffic engineering issues and real-time provisioning.

MOHAMED A. ALI (eali@ees1s0.engr.cuny.cuny.edu) received his Ph.D. degree in electrical engineering from the City College of the City University of New York in 1989. From 1980 to 1982 he was a project engineer at the Gulf Electrical and Mechanical Corporation, Cairo, Egypt. He joined the faculty of electrical engineering at the City College of New York in 1989, where he is currently a professor. His research interests cover the fields of optical device technology, high-speed optical communication systems and architectures, optical switches and high-performance IP routers, and WDM transmission and networking. His major interests are in the areas of computer simulation and modeling of high-speed communication systems. He has published more than 60 papers in professional journals and international conferences. Dr. Ali is a recipient of the NSF Faculty Career Development Award.

RUSS KURTZ (rkurtz@realtek.com) is CTO and VP of REALabs at REALTECH Systems Corporation. In this capacity, he is responsible for an organization that provides premium consulting work with leading network equipment vendors and carriers. His group is also responsible for assessing network architecture and service configurations. Kurtz received a BSEE from Purdue University in 1977 and a Doctorate in relativistic physics in 1981 from the University of Texas at Austin. He joined Bell Labs in 1982. In 1990 he assumed responsibility for evolution planning for AT&T's transport network, including office cross-connect and multiplexing equipment and fiber optic line technologies. This work resulted in AT&T's current program to comprehensively deploy SONET-based WDM line systems in ring and point-point arrangements. In partnership with then AT&T Network Cable Systems, the work helped define what is now known as Lucent's True Wave dispersion compensating fiber product. In 1995 he took over a group responsible for developing an ATM infrastructure for AT&T's SONET and packet-based data services. Kurtz joined REALTECH Systems Corporation in 1999. He has authored and co-authored public papers on electronic hardware designs and telecommunications network design.

DAN GUO [M] (dguo@turinnetworks.com) is a senior staff engineer at Turin Networks, Petaluma, CA, where he helped create the Traverse(tm) Multiservice Optical Transport Platform. Prior to joining Turin Networks, he was a manager and network architect at Sorrento Networks, San Diego, CA, where he designed Sorrento's optical channel protection protocol engine and wavelength routing software. He previously held senior engineering positions at Cascade Communications Inc., Westford, MA, and at GTE Laboratories, Waltham, MA, working on IP and ATM network design and traffic engineering. Prior to his career in industry, he was a member of the research staff in the Center for Telecommunication Research (CTR) at Columbia University, and a research investigator in the Optical Network Technology Consortium (ONTCC) funded by DARPA. He received a B.S. degree from the University of Science and Technology of China in 1987, and M.Phil and Ph.D. degrees in computer science from Columbia University, New York, NY, in 1993 and 1995, respectively. He has published more than 20 papers in research journals and international conferences.

## Errata

In the article entitled "A Survey on TCP-Friendly Congestion Control," published in the May/June issue of IEEE Network Magazine, the following table was reproduced incorrectly. The correct version of the table is published here.

Protocol	Unicast/multicast	Congestion control mechanism	Network support	Protocol complexity	Smoothness of the rate	Bias against high RTTs	TCP friendliness
<b>Single-rate</b>							
RAP	Unicast	Rate	End-to-end	Low	Sawtooth	Yes	Limited
LDA(+)	Unicast	Rate	End-to-end	High	Sawtooth	Yes	Acceptable
TFRC	Unicast	Rate	End-to-end	Medium	Smooth	Yes	Good
TEAR	Multicast	Rate	End-to-end	Low	Smooth	Yes	Good
RLA & LPR	Multicast	Window	End-to-end	Low	Sawtooth	Yes	Good
MTCP	Multicast	Window	Required	Low	Sawtooth	Yes	Good
NCA	Multicast	Window	Required	Medium	Sawtooth	Yes	Good
PGMCC	Multicast	Window	Optional	Medium	Sawtooth	Yes	Good
<b>Multirate</b>							
RLC	Multicast	Rate	End-to-end	Medium	Layer-dependent	No	Acceptable
FLUD-DL	Multicast	Rate	End-to-end	High	Layer-dependent	No	Acceptable
LTS	Multicast	Rate	End-to-end	Medium	Layer-dependent	Yes	Acceptable
TFRP	Multicast	Rate	End-to-end	Medium	Layer-dependent	Yes	Acceptable
MDLA	Multicast	Rate	End-to-end	High	Layer-dependent	Yes	Acceptable
Rainbow	Multicast	Window	Required	Low	Sawtooth	Yes	Good

■ Table 1. Characteristics of the presented congestion control protocols.

# Resonant-cavity-enhanced heterostructure metal–semiconductor–metal photodetector

Xiyang Chen and Bahram Nabet<sup>a)</sup>

*Electrical and Computer Engineering Department, Drexel University, Philadelphia, Pennsylvania 19104*

Fabio Quaranta and Adriano Cola

*Institute of Microelectronics, National Research Council (C.N.R.-I.M.E.), Via Arnesano I-73100 Lecce, Italy*

Marc Currie

*Naval Research Laboratory, 4555 Overlook Avenue, S.W., Washington, DC 20375*

(Received 24 October 2001; accepted for publication 17 February 2002)

We report a GaAs-based high-speed, resonant-cavity-enhanced, heterostructure metal–semiconductor–metal photodetector with  $\text{Al}_{0.24}\text{Ga}_{0.76}\text{As}/\text{Al}_{0.9}\text{Ga}_{0.1}\text{As}$  distributed Bragg reflector operating around 850 nm. The photocurrent spectrum shows a clear peak at this wavelength with full width at half maximum (FWHM) of around 30 nm. At resonance wavelength, a seven-fold increase can be achieved in quantum efficiency compared to a detector of the same absorption depth. The top reflector is a delta modulation doped  $\text{Al}_{0.24}\text{Ga}_{0.76}\text{As}$  that also acts as the barrier enhancement layer thus providing very low dark current values. The breakdown voltage is above 20 V. Time response measurements show rise time, fall time, and FWHM of 8.8 ps, 9 ps, and 8.1 ps, respectively, giving a 3-dB bandwidth of about 33 GHz. Combination of low dark current, fast response, wavelength selectivity, and compatibility with high electron mobility transistors makes this device especially suitable for short haul communications purposes. © 2002 American Institute of Physics. [DOI: 10.1063/1.1470224]

In addition to their important application in vertical cavity surface emitting lasers, resonant cavities have been exploited in the design of vertical photodetectors, such as p-i-n heterojunction photodiodes, Schottky barrier internal emission photodiodes, and quantum-well infrared photodetectors.<sup>1–3</sup> Resonant-cavity-enhanced (RCE) photodetectors have attracted much attention in the past few years due to their potential in solving the trade off between high quantum efficiency and high speed while, at the same time, offering spectral bandwidth filtering useful in wavelength-division multiplexing applications.<sup>4</sup>

On the other hand, the trend towards monolithic optoelectronic integrated circuits (OEIC) motivates appreciable research activity directed towards the employment of planar photodetectors, which can be easily fabricated and are compatible with the field-effect transistor (FET) process. Planar metal–semiconductor–metal photodetectors (MSM-PDs) are good candidates for such OEIC receivers.<sup>5,6</sup> The FET technology itself is strongly effected by progress in heterojunction-based devices that take advantage of reduced dimensionality regime of conduction; the high electron mobility transistor (HEMT) being a prime example. This has motivated the development of heterojunction based photodetectors that enjoy better conduction while being compatible with HEMT technology.<sup>5</sup> In particular, we have previously proposed  $\text{AlGaAs}/\text{GaAs}$  heterostructure metal–semiconductor–metal photodetectors (HMSM-PDs) that show much less dark current than conventional MSM due to both the two-dimensional electron gas (2DEG) and the effect of barrier enhancement due to the wide-gap material.<sup>7–9</sup>

A common problem with planar, as well as vertical, photodetectors is the trade off between speed and quantum efficiency; in order to achieve a fast response from photodetectors, the depleted absorption region needs to be small for reduced path length, but this results in a decreased responsivity due to small absorption depth. Resonant cavity technique offers the possibility to balance such conflict between fast speed and sensitivity.<sup>10</sup> The photodetector presented in this letter employs a vertical resonance cavity for high-speed operation, the heterojunction that forms its top mirror, however, not only doubles as a barrier enhancement layer to reduce dark current but also is modulation doped to produce an internal electric field that aids in the transport of photocarriers.

A typical RCE photodetector is made of a Fabry–Perot cavity, with a mirror on each end, whose length determines the resonant frequency. In practice, the bottom mirror consists of quarter-wave stacks of two different suitable materials forming a distributed Bragg reflector (DBR). The top mirror can be the interface between the native semiconductor and air due to their large difference of refractive index; here we use a delta-doped heterojunction to achieve better photon reflection as well as other important electronic functionalities. Figure 1 shows a simplified structure of our RCE photodetector, where  $L_1$  is the nonabsorbing barrier enhancement layer. Recirculation of photons from the top of this layer and the bottom DBR, allows a thin absorption layer  $L_2$  to be used to minimize the response time without hampering the quantum efficiency. Here quantum efficiency ( $\eta$ ) is defined as the ratio of absorbed to incident photons, which can be written as:<sup>4</sup>

<sup>a)</sup>Electronic mail: nabet@ece.drexel.edu

$$\eta = (1 - R_1)(1 - e^{-\alpha L_2}) \left\{ \frac{(1 + R_2 e^{-\alpha L_2})}{1 - 2\sqrt{R_1 R_2} e^{-\alpha L_2} \cos(2\beta L + \Psi_1 + \Psi_2) + R_1 R_2 e^{-2\alpha L_2}} \right\}, \quad (1)$$

where  $\beta L = \beta_1 L_1 + \beta_2 L_2$ ,  $R_1$ , and  $R_2$  are the top and bottom mirror reflectivities, respectively,  $\alpha$  is the absorption coefficient of the absorption layer,  $\Psi_1$  and  $\Psi_2$  are phase shifts introduced by top and bottom mirrors, and  $\beta_1$  and  $\beta_2$  are the propagation constants in these two materials. From this formula, quantum efficiency is maximized by a highly reflective DBR, i.e., when  $R_2$  is large, and when the condition  $\cos(2\beta L + \Psi_1 + \Psi_2) = 1$  is satisfied.

There are three requirements for the materials to be selected to construct the bottom mirror: large refractive index difference, lattice matching to GaAs substrates, and band gap larger than that of the active layer so that photons are not absorbed in this layer. Based on their natural material properties,  $\text{Al}_{0.24}\text{Ga}_{0.76}\text{As}$  and  $\text{Al}_{0.9}\text{Ga}_{0.1}\text{As}$  are suitable pairs to construct this DBR mirror.<sup>11</sup> The top mirror should reduce reflection from air and recirculate the photons reflected from the bottom mirror. A 55 nm layer of  $\text{Al}_{0.24}\text{Ga}_{0.76}\text{As}$  has been employed for this purpose, which also offers the following electronic properties. First, this layer lattice matches to the absorption layer, with reduced DX-center defect levels due to low Al mole fraction, while providing surface stability. Second, it enhances the Schottky barrier between metal and GaAs due to its larger band gap. Third, this layer is delta-doped to produce a 2DEG that is confined to the vicinity of the heterojunction by the conduction band discontinuity of about 0.3 eV. The last is the most important feature of this device. The confined electronic states of the quantum well at the interface as well as the electron cloud of the 2DEG have been shown to further enhance the barrier height and reduce the dark current, and thus the noise of these detectors.<sup>12</sup> This electron cloud is confined by a vertical electric field that has also been shown to aid in transport of photoelectrons.<sup>9</sup> Finally, modulation doping of this layer makes the growth

compatible with HEMT. This top AlGaAs layer is delta-doped, rather than uniformly, in order to take advantage of high channel electron density, reduced trapping effects, and improved threshold voltage as well as high breakdown characteristics.<sup>13,14</sup>

A schematic cross section of the grown RCE heterojunction MSM is shown in Fig. 2. The layer structure was grown by solid-source molecular-beam epitaxy on a semi-insulating GaAs substrate. Twenty periods  $\text{Al}_{0.24}\text{Ga}_{0.76}\text{As}/\text{Al}_{0.9}\text{Ga}_{0.1}\text{As}$  DBR were grown on a 200 nm GaAs buffer layer. The bottom mirror was designed for high reflectance at 830 nm center wavelength. The thickness of the top barrier enhancement layer is 50 nm and the spacer layer is 5 nm. A Si delta-doped layer with sheet density of  $5 \times 10^{12} \text{ cm}^{-2}$  was grown between barrier enhancement and spacer layers. The device area was  $40 \times 40 \mu\text{m}^2$  with a typical interdigital pattern with finger width of  $1 \mu\text{m}$  and distance of  $4 \mu\text{m}$ .

A transmission line model<sup>4</sup> was employed to design the layered structure and to simulate the optical properties of the RCE HMSM photodetector. The thickness of the absorption layer,  $L_2$ , is calculated by optimizing the quantum efficiency (1), which results in the condition  $2(\beta_1 L_1 + \beta_2 L_2) + \Psi_1 + \Psi_2 = 2m\pi$ . This condition is satisfied for integer values of  $m$ , with a higher number giving longer absorption length but slower response. A length  $L_2 = 117.5 \text{ nm}$  was a satisfactory trade off between speed and responsivity while still being only a fraction (11%) of the penetration depth.

Figure 3 shows the simulation results of the quantum efficiencies as a function of wavelength at two different angles of incidence. Compared to a device without a resonant cavity, a seven-fold enhancement is achieved in quantum efficiency. The peak of quantum efficiency varies with angle of incidence due to a difference in optical path length. The calculated full width at half maximum (FWHM) is around 30 nm. This value can be reduced to less than 10 nm if the reflectance of the top mirror is larger than 0.9, but that

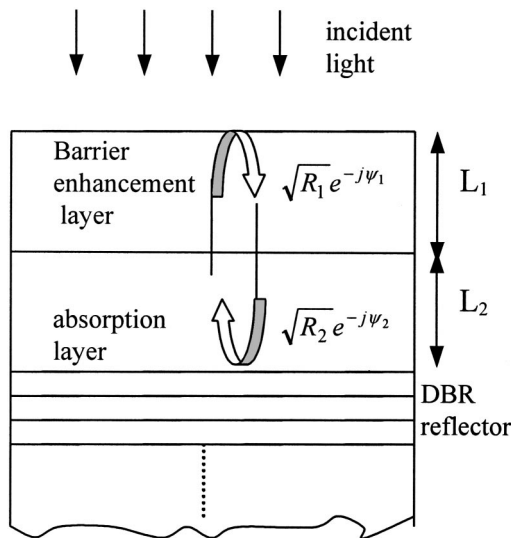


FIG. 1. Schematic diagram of the RCE heterojunction photodetector.  $R_1$  and  $R_2$  are reflectivities and  $\Psi_1$  and  $\Psi_2$  are phase shifts introduced by the top and bottom mirrors, respectively.

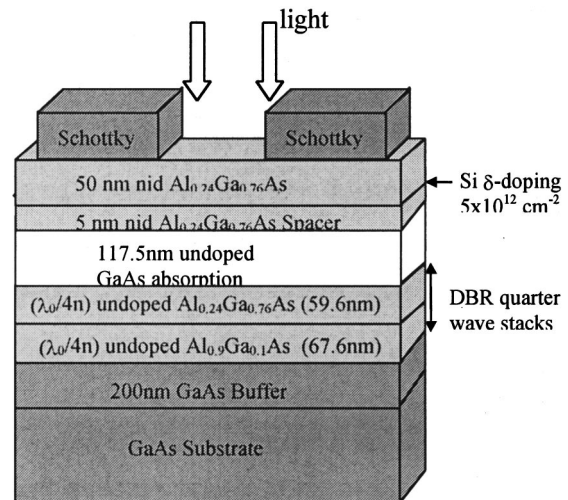


FIG. 2. Device structure of the RCE HMSM photodetector.



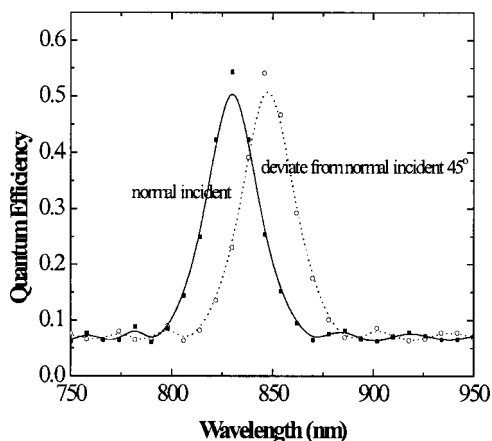


FIG. 3. Simulation results for the quantum efficiency of the layered structure as a function of wavelength for two different incident angles.

would require a much thicker layered structure.

Figure 4 shows the experimental photocurrent spectral response of the RCE-HMSM-PD. A monochromator with 0.15 nm resolution was used to select the excitation wavelength from a chopped tungsten light source. The signal was measured by a lock-in amplifier. The spectral response was measured under 10 V reverse bias. The resonant peak value is around 850 nm due to the angle of incidence of the single-mode fiber optic line, in accordance with Fig. 3. The FWHM value is seen to be in good agreement with the simulation results of Fig. 3. The shape of the photocurrent response, however, is asymmetric. This is due to the fundamental absorption edge of GaAs, which is around 870 nm and limits low energy absorption. Also, Fig. 4 is not normalized to photon flux, which would make it more comparable with Fig. 3. Finally, dependence of the absorption coefficient on wavelength is not included in the simulation. A large increase in the photocurrent is observed around 710 nm, which is due to absorption in  $\text{Al}_{0.24}\text{Ga}_{0.76}\text{As}$  layers. Dark current of the device was around 15 picoamps at this bias, which normalizes to the very low value of 9.2 femtoamps/ $\mu\text{m}^2$  of device area.

High-speed time response measurements were made us-

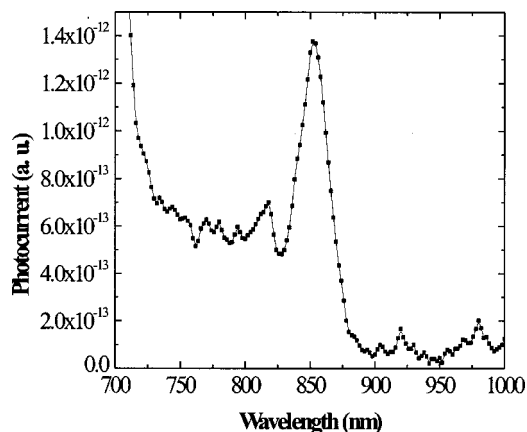


FIG. 4. Photocurrent spectral response of the RCE HMSM-PD measured at 10 V reverse bias.

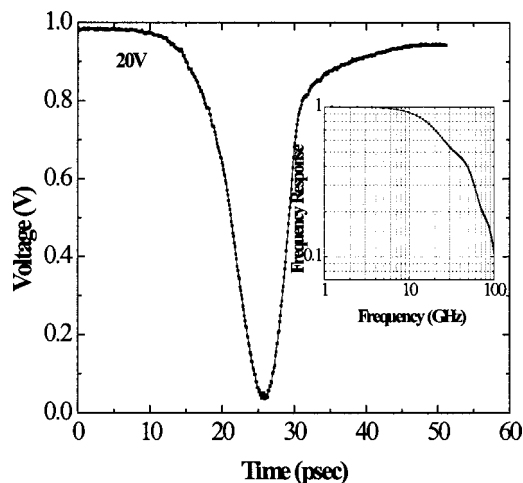


FIG. 5. Temporal response of the photodetector; insert shows the calculated frequency response.

ing a mode-locked Ti: Sapphire laser, operating at a repetition rate of 76 MHz that generated 100 fs pulses at a wavelength of 850 nm. Figure 5 shows the temporal response of a photodetector with a 1  $\mu\text{m}$  finger and 4  $\mu\text{m}$  spacing between fingers, measured by a 50 GHz sampling scope at 20 V bias. As seen in Fig. 5, FWHM of the time response is 8.1 ps, its rise time is 8.8 ps, and fall time is 9 ps. Fourier transform of the data is shown in the inset of Fig. 5 and has a 3 dB (photocurrent) bandwidth of 34 GHz.

In conclusion, we have demonstrated a resonant-cavity-enhanced, heterostructure MSM-PD operating around 850 nm. The device exhibited wavelength channel selectivity, high quantum efficiency, and high-speed characteristics. Combination of low dark current, fast response, wavelength selectivity, and integrability with HEMTs makes this device especially suitable for short haul communications purposes.

This work was partially supported by NSF ECS award 0117073 and DARPA NGI program. Wafers were grown by RJM Semiconductors Inc.

- <sup>1</sup>E. Özbay, İ. Kimukin, N. Biyikli, O. Aytür, M. Gökkavas, G. Ulu, M. S. Ünlü, R. P. Mirin, K. A. Bertness, and D. H. Christensen, *Appl. Phys. Lett.* **74**, 1072 (1999).
- <sup>2</sup>I. Kimukin, E. Ozbay, N. Biyikli, T. Kartaloglu, O. Aytür, M. S. Unlu, and G. Tuttle, *Appl. Phys. Lett.* **77**, 3890 (2000).
- <sup>3</sup>A. Shen, H. C. Liu, M. Gao, E. Dupont, M. Buchanan, J. Ehret, G. J. Brown, and F. Szmulowicz, *Appl. Phys. Lett.* **77**, 2400 (2000).
- <sup>4</sup>M. S. Ünlü and S. Strite, *J. Appl. Phys.* **78**, 607 (1995).
- <sup>5</sup>J. H. Burroughes, *IEEE Photonics Technol. Lett.* **3**, 660 (1991).
- <sup>6</sup>P. Fay, W. Wohlmuth, A. Mahajan, C. Caneau, S. Chandrasekhar, and I. Adesida, *IEEE Photonics Technol. Lett.* **10**, 582 (1998).
- <sup>7</sup>J. Culp, B. Nabet, F. Castro, and A. Anwar, *Appl. Phys. Lett.* **73**, 1562 (1998).
- <sup>8</sup>B. Nabet, *IEEE Photonics Technol. Lett.* **9**, 223 (1997).
- <sup>9</sup>B. Nabet, A. Cola, F. Quaranta, M. Cesareo, R. Rossi, R. Fucci, and A. Anwar, *Appl. Phys. Lett.* **77**, 4007 (2000).
- <sup>10</sup>Z.-M. Li, D. Landheer, M. Veilleux, D. R. Conn, R. Surridge, J. M. Xu, and R. I. McDonald, *IEEE Photonics Technol. Lett.* **4**, 473 (1992).
- <sup>11</sup>S. Adachi, *J. Appl. Phys.* **58**, R1 (1985).
- <sup>12</sup>A. Anwar and B. Nabet, *IEEE Trans. Microwave Theory Tech.* **50**, 68 (2002).
- <sup>13</sup>N. Moll, M. R. Heuschen, and A. Fisher-Colbrie, *IEEE Trans. Electron Devices* **35**, 879 (1988).
- <sup>14</sup>E. F. Schubert, *J. Vac. Sci. Technol. A* **8**, 2980 (1990).

# **Experimental Field Trial of Waveband Switching and Transmission in a Transparent Reconfigurable Optical Network**

**Paul Toliver, Robert J. Runser, Jeffrey Young, and Janet Jackel**

*Telcordia Technologies, 331 Newman Springs Rd., Red Bank, NJ 07701*

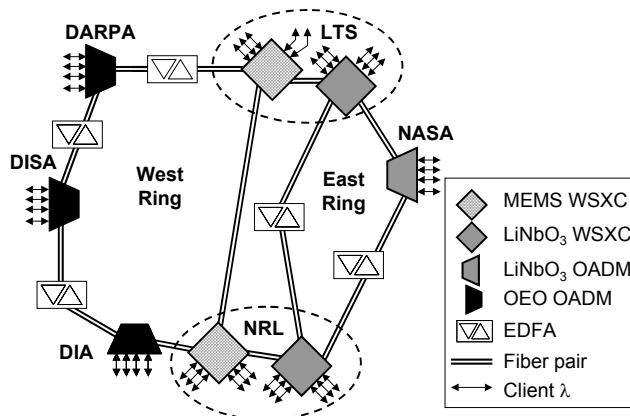
*Tel: (732) 758-3057; Fax: (732) 758-4372; email: ptoliver@research.telcordia.com*

**Abstract:** We demonstrate 4-channel, 25GHz-spaced waveband transmission in a 200GHz passband through a transparent reconfigurable optical network. We investigate the impact of polarization effects on the waveband and its interaction with MEMS and LiNbO<sub>3</sub> switch fabrics.

## Introduction

Waveband switching is a technique in which a subset of wavelengths are grouped together and switched optically as a single entity through a transparent network infrastructure. The individual wavelengths of a waveband are commonly assumed to be contiguous in spectrum, although they could also be interleaved or arbitrarily spaced. One of the key advantages of switching groups of wavelengths that share a common multi-hop path rather than individual wavelengths is that fewer ports are required on the switch fabrics of photonic cross-connects at transit nodes. Wavebands can also be used to upgrade the capacity of existing transparent networks without requiring any changes to the core network elements provided the waveband signals generated at the edge fit within the allowable optical passbands. This technique can also be used to support higher channel bandwidths on fiber paths that would otherwise require costly impairment compensation or regeneration.

The benefits of waveband switching have been studied by a number of groups from an architectural perspective [1, 2]. To date, however, there have been few reports discussing the practical implementation issues over real networks. Here, we present the results of waveband switching experiments that were demonstrated on the Advanced Technology Demonstration Network (ATDnet)—an optical network testbed that links a number of government agency laboratories in the Washington, D.C. metropolitan area (see Fig. 1). Under the MONET program, the original ATDnet testbed was designed to support only 8 x 2.5 Gb/s wavelengths per fiber [3]. The infrastructure of the East Ring supports network elements (NEs) that are optically transparent enabling the possibility for considerably higher bandwidths but with the need for external dispersion compensating elements. For the particular results summarized in this paper, the waveband switching experiments primarily utilized paths through the MEMS and LiNbO<sub>3</sub>-based transparent wavelength selective cross-connects (WSXC) located at both the Laboratory for Telecommunication Sciences (LTS) and the Naval Research Laboratory (NRL).



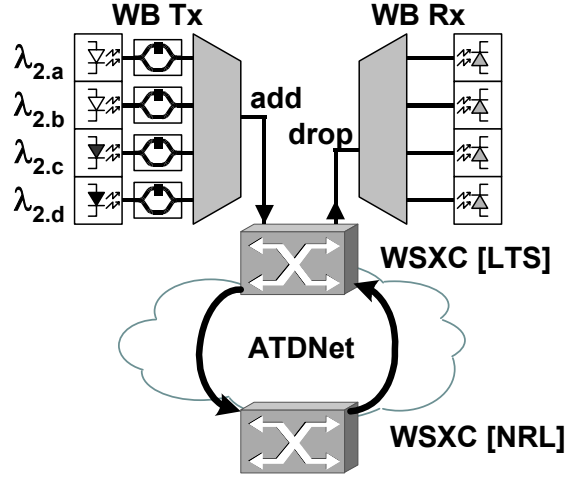
**Fig. 1.** ATDnet testbed comprised of both transparent and non-transparent network elements.

## Experiment

The transparent network elements within ATDnet are capable of supporting at least 8 wavelength passbands on a 200 GHz ITU frequency grid. We performed our waveband switching experiments using the second 200 GHz window, which is centered at 1550.92 nm, while other single wavelength signals, such as OC-48 ATM, populated the remaining seven spectral windows.

As illustrated in Fig. 2, a 4-channel waveband transmitter (WB Tx) and a 4-channel waveband receiver (WB Rx) were constructed and connected to the add/drop ports of a WSXC

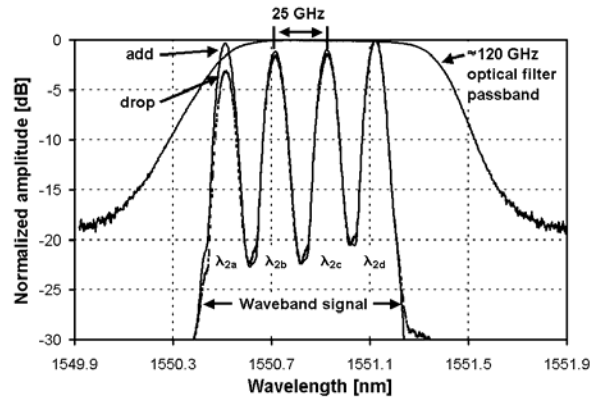
located at the LTS. The WSXC allowed us to switch the waveband signal through various transparent paths of ATDnet. The four sub-channels of the WB Tx, which consists of an array of tunable lasers followed by an array of electro-optic modulators, are passively multiplexed together on a 25 GHz grid and their amplitudes are equalized before the WSXC add port. The WB Rx consists of a cascaded array of fiber Bragg grating filters (each 15 GHz wide) and circulators along with an array of four photoreceivers. Since the ATDnet NEs are optically transparent, no changes or hardware upgrades were required to support the waveband client signal. Such an upgrade would have been impossible given an OEO switched network.



**Fig. 2.** Experimental setup for waveband switching demonstration.

## Results

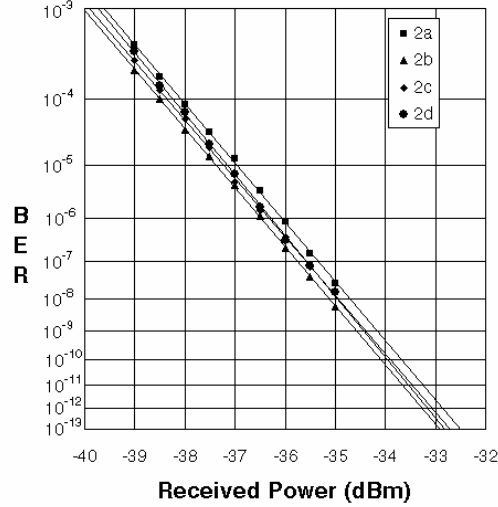
In Figure 3, we show the 4-channel waveband signal contained within a spectral window that is approximately 120 GHz wide and centered at 1550.92 nm. This particular window is defined by the optical mux and demux components inside the MEMS-based WSXC. As shown by the “add” signal plot, the four waveband sub-channels are set to the following wavelengths:  $\lambda_{2a}=1550.52$ ,  $\lambda_{2b}=1550.72$ ,  $\lambda_{2c}=1550.92$ , and  $\lambda_{2d}=1551.12$  nm. The waveband was transmitted over approximately 100 km of SMF-28 fiber on a loopback path between the LTS and NRL through MEMS-based WSXCs located at each node. The entire waveband is received at the drop port, although the  $\lambda_{2a}$  sub-channel is attenuated slightly due to its proximity to the passband edge of the NE muxes and demuxes.



**Fig. 3.** Plot of add/drop waveband spectrums and passband spectral window.

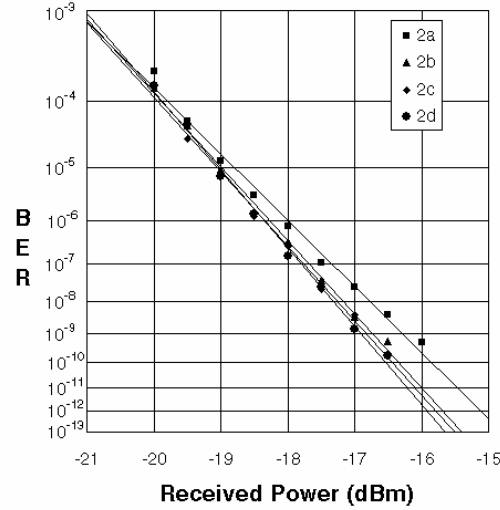


The recovered bit error rates (BERs) of all four sub-channels after transmission through the two WSXCs, 100 km of fiber, and optical amplifiers (integrated into the NEs) is shown in Fig. 4. For these BER results, the waveband sub-channels are modulated with decorrelated OC-48 (2.488 Gb/s) pseudo-random data streams ( $2^{23}-1$ ). The average receiver sensitivity given a BER of  $10^{-9}$  is  $-34.4$  dBm. Compared to back-to-back operation, the power penalty for the four sub-channels is only 0.1 dB. This experiment demonstrates that an existing transparent network infrastructure originally designed for 2.5 Gb/s capacity per passband can support 10 Gb/s per passband using wavebanding techniques at the edge of the network without requiring changes to core network elements or transmission fibers.



**Fig. 4.** Bit error rate performance with four 2.5 Gb/s channels in waveband.

We also investigated the ability to modulate the individual sub-channels at 10 Gb/s, which provides a total passband capacity of 40 Gb/s within the transmission window. In this experiment, all four wavelengths were modulated by through a common 10 Gb/s electro-optic modulator. Pre- and post-dispersion compensating fibers were used to support the 10 Gb/s sub-channels over the 100 km SMF-28 network span. The resulting BER performance for the four 10 Gb/s sub-channels is summarized in Fig. 5. The average receiver sensitivity at a BER of  $10^{-9}$  is  $-17.8$  dBm. The average power penalty across the four sub-channels was 1.1 dB, with  $\lambda_{2a}$  having the largest penalty of 1.5 dB. The increased power penalty is likely to be a result of the channel proximity to the passband edge causing greater signal distortion due to filter dispersion. It would not have been possible to achieve the transmission of a single 40 Gb/s wavelength within an ATDnet passband without subsequent compensation for PMD.



**Fig. 5.** Bit error rate performance of four 10 Gb/s sub-channels of the waveband.

### Impact of polarization effects

An important physical layer issue to consider in a waveband switched network is the impact of how impairments may impact individual sub-channels of the waveband differently. In particular, polarization effects such as polarization dependent loss (PDL) can be particularly important for transparent NEs that use optical switch fabrics such as LiNbO<sub>3</sub>. The ATDnet contains NEs with both MEMS and LiNbO<sub>3</sub> fabric technologies. The PDL for the loopback path for the MEMS-based WSXCs was less than 0.5 dB for waveband sub-channels  $\lambda_{2b}$ ,  $\lambda_{2c}$ , and  $\lambda_{2d}$  and did not vary significantly based on the launched polarization orientation of the subchannels. The first sub-channel ( $\lambda_{2a}$ ) had a higher PDL value of 1.0 dB due to its proximity to the passband edge. In contrast, when the signal was sent through the LiNbO<sub>3</sub>-based WSXC, the PDL varied within the range of 4.0-5.5 dB across the waveband. The amount of PDL experienced by each sub-channel was a function of the launch polarization orientation.

Ensuring that the sub-channels of a waveband maintain the same amplitude is important for minimizing the waveband transmission penalty. Over longer spans and multiple hops through optical networks, the amount of PDL through a network element should be kept to a minimum especially since it can coupled with second order PMD (which introduces wavelength dependent depolarization [4]) to cause additional variation in the sub-channel amplitudes.

### Conclusion

Wavebanding is a technique that can be applied to existing transparent optical elements to increase the spectral efficiency within a passband without requiring costly impairment compensation or upgrades to photonic cross-connects. It can also be applied to new network designs in order to minimize the number of switch fabric ports. We have demonstrated 4-channel waveband transmission over 100 km of legacy fiber in ATDnet within a 200 GHz optical passband with aggregate capacities of 10 and 40 Gb/s. Understanding the impact of impairments and optical nonlinearities that may affect the individual sub-channels of a waveband is important for determining the ultimate reach, capacity, and channel spacing that can be achieved within a passband of existing networks and fiber infrastructure.

This work has been supported in part by the LTS and also by DARPA under contract F30602-00-C-0167. The authors gratefully acknowledge Matthew Goodman of Telcordia Technologies and Scott McNown of the LTS for valuable technical discussions and support.

## References

- [1] L. Noirie, M. Vigoureux, and E. Dotaro, *OFC 2001*, paper TuG3, March 2001.
- [2] R. Lingampalli and P. Vengalam, *OFC 2002*, paper ThP4, March 2002.
- [3] W. T. Andersen, J. Jackel, G.-K. Chang, et al., *J. Lightwav. Technol.*, vol. 18, no. 12, pp. 1988-2009, Dec. 2000.
- [4] C. D. Poole, N. S. Bergano, R. E. Wagner, and H. J. Schulte, *J. Lightwav. Technol.*, vol. 6, no. 7, pp. 1185-1190, July 1988.

## **Design and Implementation of Software for Assembly and Browsing of 3D Brain Atlases**

Carl Gustafson<sup>1</sup>, Oleh Tretiak<sup>2</sup>, Louise Bertrand<sup>1</sup> and Jonathan Nissanov<sup>1\*</sup>

<sup>1</sup>Department of Neurobiology and Anatomy, Drexel University College of Medicine, 2900 Queen Lane, Philadelphia, Pennsylvania, USA

<sup>2</sup>Department of Electrical and Computer Engineering, Drexel University, 3141 Chestnut Street, Philadelphia, Pennsylvania, USA

### **Abstract:**

Visualization software for three dimensional digital brain atlases present many challenges in design and implementation. These challenges include the design of an effective human interface, management of large data sets, display speed when slicing the data set for viewing/browsing, and the display of delineated volumes of interest (VOI). We present a software design, implementation and storage architecture that addresses these issues, allowing the user to navigate through a reconstructed volume quickly and smoothly, with an easy-to-use human interface. The software (MacOStat, for use with Macintosh OS) allows the user to rapidly display slices of the digital atlas at any arbitrary slicing angle, complete with delineated VOIs. The VOIs can be assigned colors of the user's choosing. The entire atlas, or selected portions, may be resliced with slices stored as individual image files, complete with delineations. These delineations may be transferred to corresponding sections of experimental materials using our analysis program (Brain). The software may be obtained from the laboratory's web site: <http://www.neuroterrain.org>

### **Keywords:**

3D brain atlas, 3D reconstruction, 3D visualization

### **\*Corresponding Author:**

Tel: 1-215-991-8410; Fax: 1-215-843-9367

## Introduction

Neuroanatomical atlases are widely used to guide delineation of stained sections, assist in electrode placement, and aid in surgical planning. Numerous atlases are available [e.g. 1,2,3,4]. In general these atlases are made of a series of non-consecutive 2D images of stained sections along with graphical outlines of standard nuclei and tracts; they are usually provided as printed manuscripts. These atlases suffer two major deficiencies: they support only a limited number of orientations—typically offering coronal, sagittal and horizontal views, often incomplete (for example, in rodent atlases, the olfactory bulb is often not included in horizontal or sagittal views)—and they are sparse with substantial gaps present between sections. Three-dimensional digital atlases can overcome these deficits and a number are now available for the rat [5], human [6,7], mouse [8], rhesus monkey [10], fly [11] and other species.

To make use of these digital atlases, visualization software is required. There are generalized volume visualization applications available via both commercial (Voxblast [12], Amira [13] IDL [14]) and public (Open Visualization Data Explorer [15]) licenses; however, these are not particularly suitable for atlas display. While they do offer both voxel and vector display and do support arbitrary plane of view, all necessities in this specific setting of 3D visualization, they lack support for other important functions of atlases such as display of stereotaxic coordinates display, and textual annotations. They also fail to provide a straightforward means of matching experimental sections to equivalent atlas planes, natural navigational tools, or accommodation for the large size typical of digital atlases. We designed an application (MacOStat) that overcomes these deficits.

## Design Considerations

The primary design consideration in developing MacOStat was usability. By this is meant both an intuitive human interface, with directly manipulatable interface elements, and sufficient processing speed to make live navigation possible. An intuitive interface is of critical importance in any computer software but it is especially important in software attempting to map three dimensional data into a two dimensional display space; the third dimension must be handled in a consistent and predictable manner, and the available commands for manipulating the display should be direct and obvious to the novice user. However, there is currently no widely-accepted standard to handle these requirements. Most applications rely on a series of orthogonal views showing the projection of the slicing plane on the volume, along with a view normal to that plane [16]. This type of interface is conceptually simple - one can easily specify

the slice location. Unfortunately, live navigation using this interface is not easy — an obstacle to useability with this scheme is that manipulation is not direct — it is necessary to adjust settings in several separate windows to navigate the atlas, while monitoring the result in yet another distinct window.

Speed is the other side of the usability coin - as even moderate-resolution atlases are quite large (a 17.9 $\mu$ m/pixel isotropic mouse brain atlas [9], without any delineation, can easily exceed 100MB), efficient processing is critical to providing a satisfactory user experience. It is difficult, if not impossible, to accurately navigate through an atlas unless refresh rates are substantially faster than a frame per second. Handling this massive amount of data, and selecting and displaying individual slices quickly is another primary challenge. In addition, both fine- and coarse-grained movement is required - fine grained movement alone quickly becomes tedious when traversing large distances, and coarse-grained control makes precise positioning impossible.

To meet these criteria, we developed a system that eliminates the entire orthogonal-view paradigm, relying instead on only showing a slice of the atlas for fine-grained navigation, and a wire-frame representation of the slice's location within the atlas' bounding volume for coarse-grained movement. These displays are directly manipulated - the user selects an action from a toolbar, then clicks and drags the mouse over the display; the display moves in direct response to the user. To support rapid display and efficient access to atlas data, we have developed a novel data storage technique that is generally applicable to volume (voxel) data.

A second design consideration is an architecture that facilitates maintenance and enhancement. It is inevitable that software applications will need fixes and additional features. This is especially true in relatively new content domains, such as interactive digital atlases. Unless the basic program architecture is initially well factored, and modular in nature, future attempts at enhancements will result in increasing fragility, resulting in either costly redesign, or a reluctance to add additional features.

To realize this goal, we use object-oriented design techniques, and a commercial application framework, MacApp [17]. The framework uses a Model-View-Controller (M-V-C) design pattern well-adapted to user-centric interface design, and is easily modifiable, via inheritance techniques, to our content domain. Atlas data is abstracted into generic voxel maps, with appropriate metadata handled by both customized framework and purpose-built classes, acted upon by generic slicing classes. A hierarchy of command classes abstract user interactions. Customization of framework view classes provides the display functionality. An additional

advantage of object-oriented design is that it facilitates reuse of existing code bases. In our case, we have an image acquisition and analysis package (Brain [18]), developed in-house, that we use for 2D delineation of the data sets used in atlas creation. Many of the classes related to the data model in Brain have been reused in the development of the atlas browser. The reuse already developed and debugged code greatly shortens the application development cycle.

## System Overview

We use an M-V-C pattern for our system. All the data structures (the model) that comprise an atlas (gray-scale data, delineation, size and calibration, metadata, etc.) are aggregated by a class descended from the MacApp base class TDocument (Figure 1). The atlas is displayed (the view) using classes derived from the MacApp base class TView (Figure 2). Finally, all interactions with the user (the controller) are handled by derivatives of the base class TCommand (Figure 3). Finally, we have developed an additional class hierarchy independent of the MacApp framework used to handle data slicing for later display (Figure 3).

### The Model

The atlas browser has two independent components — the program to construct atlas data files from separate, pre-aligned frames (Glucose), and the browser itself (MacOStat). To the greatest extent possible, the two share the classes that implement the data model (the class hierarchy from the atlas browser is shown in Figure 1). Both also share classes with our analysis package, Brain; atlas data files are constructed from Brain image files, and the atlas browser must export resliced images in Brain file format. This means that our root data model has two independent branches, one for the atlas document and one for the image documents that can be matched to the atlas. The document objects used to construct an atlas (not shown in Figure 1) contain additional data elements beyond those required by the Brain application— for example, the position and orientation of the image within atlas space. To accommodate this, we created a subclass of the basic Brain document to hold orientation data. All document objects have methods to read and write data files when appropriate; this allows us to use existing classes, and add additional capabilities via inheritance. Part of the initialization for document classes usually includes the creation of view objects to display the document's data. Because this display is both not needed and computationally expensive when constructing an atlas from images, we further subclass the document to eliminate this view creation step.

In addition to the document classes based on code developed for Brain, we also have been able to reuse classes developed to represent delineated regions of interest and calibration curves.

Added to this are 3D specific classes to represent both gray-scale and binary volume data (image voxels and volumes of interest) and metadata, as well as various collection classes used to manage these data objects, which were derived from framework collection classes.

## **The View**

Previous work provided a view class that used an offscreen buffer for drawing; this class was used to display individual image files opened for matching purposes. A similar class was developed to handle display of the sliced atlas image. Additional classes were derived from framework classes for tool and informational views, and so forth.

Most views are subclassed directly from the base framework view class, but some intermediate classes are used, primarily in the views that can be manipulated via mouse (Figure 2). In `TMouseOrientableView`, the code that responds to mouse actions resides in a superclass of the final view objects, as all views that can be manipulated via mouse (wire frame coarse-positioning and slice fine-positioning views) may then use common code.

Many views are designed to be embedded in other views, to control various aspects of view behavior. For example, `T3DBufferedView` is the ultimate destination for sliced atlas images and the associated sliced volumes of interest. It uses an offscreen image buffer to assemble the final image prior to display. This view is nested into a view descended from `TMouseOrientableView`, which controls responses to mouse movement. While it is possible to derive `T3DBufferedView` from `TVolumeView` or `TMouseOrientedView`, we also allow multiple views to be nested in `TVolumeView` - for example, it is possible to browse several different data sets, but keep all data sets at the same exact slicing orientation. By separating the view classes, we can allow `TVolumeView` to handle both tracking and scrolling of the views, and transfer information to all instances of the data model; the code to handle this if the hierarchies were combined would be significantly more complex.

## **The Controller**

A number of command classes were derived from the framework command hierarchy to encapsulate user actions such as navigation and reslicing. One advantage to this structure is that each command object contains the information needed to undo any actions; commands may be placed on a stack for multiple levels of undo if desired. By supporting undo for almost all user actions, the human interface becomes much more forgiving, allowing the user the confidence that any mistakes can be recovered.



In addition to the suite of command classes, a set of classes used to reslice volume data was created. In essence, there is one slicer class for each type of volume data (gray-scale and binary) and display variant (8 or 32 bit pixels, for example). These slicers, which are persistent objects, are controlled by the command objects spawned by the user's actions. In this case, much of each slicer's code is independent of the other slicers in the suite, primarily for performance reasons - the overhead of calling virtual functions (the basic mechanism allowing a subclass to provide specialized behavior) is normally negligible, but in the tight loops used in slicing, it has the potential to become significant.

## **File Structure**

Atlas files are stored in a tagged format. Each data item in the file is preceded by a tag identifying the data, and by a length field specifying how long the field is. This structure provides several benefits; the first is position independence — each chunk of data is interpreted independent of the data around it, and is not required to appear at a particular offset in the disk file — it may be inserted anywhere in the file. The second advantage is that the file structure is effectively separated from the program's version. A file created by a later version of the atlas construction software, Glucose, may still be read and understood by an earlier version of MacOStat, it simply skips unrecognized tags. This in turn means that development and distribution of atlas construction and browsing software can proceed independently. A common problem in many applications is the lack of forward file compatibility, this block structure eliminates the problem in all except the most extreme cases, and thus allows the developers more freedom in adding useful features.

## **Implementation Details**

Data structures, either in the form of classes or other formalisms, make up the core of an application. A well-designed data structure must model the reality being represented and must provide a structure easily accessible to, and supportive of, the internal program code. Object-oriented design methods (in which classes include both the data and the functions to manipulate that data) are often used to meet these requirements. Good classes present an opaque interface to the application allowing access to the data the instantiated objects maintain, while hiding the actual implementational details from the application. This data hiding results in better modularity, which allows future enhancements to be made without disrupting the existing the program code.

## **Macrovoxels**

As an example of the value of opaque data structures, let us first consider the method we use for storing atlas image voxels. A typical method of organizing volume data is in the form of a monolithic array of voxels; we do not use this structure for reasons that will be detailed later. Instead, we group voxels into clusters we call macrovoxels, and organize these macrovoxels into a structure we refer to as a voxel map (Listing 1, 2). To access an individual voxel, one requests a specific macrovoxel by providing the voxel map with the macrovoxel's address in atlas space, and then requests the individual voxel by providing that voxel's address to the macrovoxel. A voxel value is returned. At no point does the voxel map need to understand how the individual voxel data is stored, so macrovoxels supporting binary, 8, 12, 16 or wider bit data may be stored in the voxel map. In practice, the voxel map's client needs to know the bit dimensions of the individual voxels, as it needs to set the correct pixels in the screen buffers for display, but no other part of the application needs to be aware of this implementation detail.

We use the macrovoxel structure for both performance and storage considerations. Computer memory and file systems are essentially one-dimensional systems, where the only access to an individual data item is by some index from a defined location. Mapping multi-dimensional data into memory is essentially an exercise in bookkeeping, however, locality of data is preserved only along one dimension - if the voxel at  $[X,Y,Z]$  is adjacent to the voxel at  $[X+1,Y,Z]$ , and the maximum dimensions of the atlas are  $(m,n,o)$ , then the distance to  $[X,Y,Z+1]$  is  $m \times n$  voxels away. This is an important consideration when accessing large data sets for two reasons. First, modern processors use a cache to provide quick access to recently used data, and nearby data - a fixed amount of data is moved into a cache (a cache line), on the first access, and after this, the cached copy of the data is accessed, rather than the data in RAM. Cache memory access is much faster. If the data to be accessed during slicing is localized, most of it can be loaded into cache, and processing speed enhanced. If it isn't cached, then the "cache hit rate" drops, and processing slows down. Second, modern systems utilize a virtual memory manager (VMM), which trades disk storage which is slow but capacious, for RAM memory, which is more limited in capacity but much faster. When a particular memory location that has been paged to disk is required, the VMM brings it into RAM memory. This is a slow process, and so it is desirable to minimize these page loads.

The macrovoxel architecture reduces some of the drawbacks of large monolithic arrays. We group nearby voxels into a chunk, typically 16 voxels on edge. These chunks are then arranged into a structure we call a voxel map (Figure 4). The voxel map comprises the atlas data, and is the only entity to which the document class needs to maintain a reference. By grouping voxels into these macrovoxels, we now have a data block that is roughly the size of a VMM page, and

also fits into a small number of cache lines. Thus, any voxels (row, column, plane) that are spatially adjacent are now also physically adjacent in memory, which improves memory access performance. Consider an attempt to reslice an atlas: if the slicing plane is coincident with the data set (i.e., examining only a single plane of data in an array) then only the data to be displayed, plus possibly a cache line and/or VMM page frame on either side of the data plane need to be moved. In this case, slicing the monolithic array will be faster than a macrovoxel array, as in the latter 16 planes (based on a 16-voxel on edge macrovoxel) will need to be moved. If, however the slicing plane is perpendicular to the data set major axis (cuts across many array planes), then given the nature of the array, much more data will need to be moved for the monolithic array than for a macrovoxel array (Figure 5).

An additional advantage of the macrovoxel architecture is that it allows us to eliminate empty voxels from storage. By empty voxels we mean those voxels that exist outside the contour of the 3D image we are representing. Biological images are rarely in the form of cubes or prisms, but that is how most arrays (including macrovoxel arrays) represent data - this representation means that location information may be computed given the three dimensions of the array, rather than stored in some fashion. The result is a reasonably compact data representation. This fails, however, if the object being represented deviates significantly from a prism. In our domain, brain imaging, the subject matter more closely approaches an ovoid. This means that the voxels in the corner of the atlas volume have no useful data in them. By breaking our atlas space up into macrovoxels, we can identify macrovoxels that contain these empty voxels, and replace that macrovoxel's entry in the volume map with a reference to a single empty or "white" macrovoxel. This allows us to eliminate large amounts of empty space, resulting in a reduced disk and memory footprint, with a savings of typically 30%. (Figure 6)

## **Slicing**

The actual slicing (Listing 3) is handled by taking the corners of the virtual knife (the plane to be displayed on screen), and converting from an atlas-based coordinate system, where the axis units are arbitrarily defined relative to the atlas geometry, to a coordinate system where the units are based on macrovoxel indexes. Atlas-based coordinates are centered in atlas space, which facilitates the rotation and translation of the virtual knife, while macrovoxel-based coordinates are based on the upper front left macrovoxel, to facilitate memory accesses. This conversion results in the useful property that the macrovoxel containing any given voxel is specified by the whole number portion of the voxel's coordinate. The fractional part is then scaled by the macrovoxel size (usually 16 voxels) to give the exact voxel. This scaling thus allows voxel address computation almost as efficiently as if a monolithic array were to be used.

Once this scaling is completed, we divide opposite edges of the virtual knife into segments corresponding to one of the dimensions of the display space. From each of these segments, we project transect vectors to the opposite side of the virtual knife, and divide these into segments corresponding to the other dimension of display space. Each of these segments now can be mapped directly to a voxel in our macrovoxel array, and the image buffer is populated by iterating over the segments on the virtual knife (Figure 7).

Finally, we provide a class to specify the virtual knife location. Any changes to slicing angle or position are forwarded to this class, to which the slicing classes refers. Using this mechanism, the slicing classes need no knowledge of the basic document objects - they only need to know about the data set to be sliced, the screen buffer the sliced image is written to, and the virtual knife location.

### **Atlas construction**

Atlas construction (via Glucose) is essentially the reverse of atlas slicing. First, an array of macrovoxels is created. Here, the entire array needs to be allocated, as the final population will not be known until construction is complete. Each pre-aligned frame comprising the data set is loaded, it's location in atlas space determined either by data included with that file or by it's position in the list of frames, and it's pixels converted to voxels and written to the macrovoxel collection. During this process, gray values are equalized or remapped based on any included calibration curves. Once all frames have been loaded, the collection of macrovoxels is checked for data content, and empty macrovoxels eliminated. The completed data set is then written to disk. Delineations, in the form of outlines, are accumulated and handled in a similar fashion. The final calibration curve, a table of VOI names and display data, and coordinate system transformation data are also written to file.

## **Human Interface**

The human interface of MacOStat provides the user with a view of the data set at the current virtual knife position and any associated delineation, a schematic view of atlas space showing the position of the virtual knife, and the necessary controls and menus to control the application (Figures 8,9).

Using a control strip,(a utility window with icons representing commands) the user can switch basic slicing axis or presentation: coronal, sagittal, or horizontal. Rotation and translation of the virtual knife is also controlled by this strip.

Navigation through atlas space is done by translating and rotating the virtual knife. These actions are carried out via mouse movements: The user selects the action desired from the navigation control strip, then clicks and drags the mouse over either the view of the atlas slice, or over the wire frame diagram of the virtual knife and atlas space. Dragging over the wire frame provides coarse movement control, and the slice view provides fine movement control. In both cases, the display content changes to provide direct feedback to the user.

In addition, the rotation and translation of the virtual knife is provided in the upper left corner of the wire frame window. In addition, a set of text entry boxes are provided to allow the user to specify these values numerically. Finally, a facility is provided to remember particular virtual knife orientations, and recall them via a popup menu. These saved positions can may be written to file, and loaded into memory.

Dragging the mouse over the atlas slice display itself also displays the X, Y and Z coordinates of the mouse, expressed in atlas units. Data needed for conversion of atlas coordinates to stereotaxic coordinates can be embedded in the atlas data files, and will be used if available. In this case, the mouse pointer location is expressed in stereotaxic coordinates rather than the atlas-relative coordinates.

Atlas data sets may also contain delineation data; this data is shown either as a colored outline on the gray-scale image, or as a translucent colored area. Colors are user-selectable. In addition, the user may turn the delineation display on or off, and may define a subset of individual structures for display (Figure 9). Display lists complete with color specifications may be saved to a file, and reloaded as desired. The file itself is in the form of tab-delimited text, and so may be viewed and edited with standard word processing and spreadsheet applications. Each structure is specified by both an abbreviation and a name.

Finally, it is possible to save sequential virtual slices to individual data files. To save these slices, the starting and stopping positions are marked, and the number of slices and manner in which the virtual knife position is interpolated is specified. For example, if the starting and stopping slices are not parallel, one can specify that only the starting or stopping knife angle should be used, the mean angle, or that the angle should be interpolated across the span. Using this facility, the user can open individual experimental sections and match them to the 3D atlas to rapidly generate a set of matching planes that can then be employed in delineation of the experimental data using Brain.

## **Software Availability**

The software (MacOStat) may be downloaded without fee from [www.neuroterrain.org](http://www.neuroterrain.org). It requires MacOS 8.6 and CarbonLib 1.3 or later.

## **Acknowledgment**

This work was supported by NIH award P20 MH62009 and US Air Force agreement F30602-00-2-0501

## References

- [1] Franklin, B. J. Keith, G. Paxinos, The Mouse Brain in Stereotaxic Coordinates. Academic Press, San Diego, California 1997
- [2] Paxinos, G., C. Watson, The Rat Brain in Stereotaxic Coordinates. Academic Press, San Diego, California 1998
- [3] Swanson, L. W., Brain Maps: Structure of the Rat Brain. Elsevier, Amsterdam, Netherlands, 1992
- [4] Mai, J. K., Assheuer, J., G. Paxinos, Atlas of the Human Brain. Academic Press, San Diego, California 1997
- [5] Toga, A. W., E. M. Santori, R. Hazani, K. Ambach. Rat Atlas Image Database, [http://www.loni.ucla.edu/Research\\_Loni/atlas/rat/](http://www.loni.ucla.edu/Research_Loni/atlas/rat/)
- [6] Sundsten, J W. Digital Anatomist: Interactive Brain Atlas, <http://www9.biostr.washington.edu/da.html>
- [7] Kikinis, R. A DIGITAL BRAIN ATLAS FOR SURGICAL PLANNING, MODEL DRIVEN SEGMENTATION AND TEACHING, <http://www.spl.harvard.edu:8000/pages/papers/atlas/text.html>
- [8] Sidman, R. L., B. Kosaras, B. Misra, S. Senft. High Resolution Mouse Brain Atlas, <http://www.hms.harvard.edu/research/brain/>
- [9] Bertrand, L., J. Nissanov, 3D Atlas of the Mouse Brain, Computer Vision Laboratory for Vertebrate Brain Mapping, Philadelphia, 2001. <http://www.neuroterrain.org/>
- [10] Jones, E G. et al, (Resus atlas) UC Davis/UC San Diego Human Brain Project, <http://neuroscience.ucdavis.edu/hbp/project2.html>
- [11] Flybrain, <http://flybrain.neurobio.arizona.edu/Flybrain/html/>
- [12] VoxBlast <http://www.vaytek.com/VoxBlast.html> (commercial)
- [13] Amira -- visualization and reconstruction for 3D image data. <http://www.amiravis.com> (Commercial)
- [14] IDL -- data analysis, visualization and application development. <http://www.rsinc.com/> (Commercial)
- [15] Open Visualization Data Explorer -- an application and development software package for visualizing 2D/3D data. <http://www.research.ibm.com/dx/> (IBM Public License)

- [16] Lohmann, K., Gundelfinger, E. D., Scheich, H., Grimm, R., Tischmeyer, W., Richter, K., Hess, A. (1998) BrainView: a computer program for reconstruction and interactive visualization of 3D data sets. *J Neurosci Methods* **84(1-2)** 143-154.
- [17] Apple Computer, Inc., MacApp, <http://developer.apple.com/tools/macapp/>
- [18] Nissanov, J., D.L. McEachron. 1991. Advances in image processing for autoradiography. *J. Chem. Neuroanat.* 4:329-342.



CARDIOMYOCYTE-MEDIATED CONTACT PROGRAMS HUMAN  
MESENCHYMAL STEM CELLS TO EXPRESS CARDIOGENIC PHENOTYPE

Sunil Rangappa, M.D.

John W.C. Entwistle, Ph.D., M.D.

Andrew S Wechsler, M.D.

J. Yasha Kresh, Ph.D.

Department of Cardiovascular Medicine & Surgery, Drexel University College of  
Medicine, Philadelphia, PA

Address correspondence to:

J. Yasha Kresh, Ph.D.  
Dept. of Cardiovascular Medicine and Surgery  
Mail Stop 111, 245 North 15th. Street  
Drexel University College of Medicine  
Philadelphia, PA 19102

*Abstract was accepted for oral presentation at the American Heart Association's  
Scientific Sessions, Chicago, Illinois, November 17-20, 2002.*

*Acknowledgment:*

This study was supported in part by State of Pennsylvania (#11700014), Defense Advanced Research Agency (DARPA), and Grant-in-Aid from Cardiovascular Institute of Philadelphia, PA.

## Abstract:

**Background:** Intercellular cross talk and cellular plasticity are key factors in embryogenesis and organogenesis. The microenvironment plays a critical role in directing the progression of stem cells into differentiated cells. We hypothesized that intercellular interaction between adult human mesenchymal stem cells (hMSC) and adult human cardiomyocytes would induce stem cells to acquire the phenotypical characteristics of cardiomyocytes, and tested the role that direct cell-to-cell contact plays in directing this differentiation process. hMSCs were cultured in the presence of human cardiomyocytes ("co-culture"), or in the presence of media conditioned by separate cultures of human cardiomyocytes ("conditioned media").

**Methods:** Human cardiomyocytes were labeled with chloromethyl derivatives of fluorescein diacetate (CMFDA). In the co-culture experiments, hMSCs and human cardiomyocytes were mixed at a 1:1 ratio in Sm2 media and seeded at cell density of 10000 cells/cm<sup>2</sup>. Cells were co-cultured in an incubator at 37<sup>0</sup>C for 48 hrs. Subsequently, Fluorescence Activated Cell Sorting (FACS) was used to extract the differentiating hMSCs. In the conditioned media experiments, hMSCs were incubated in media previously conditioned by cardiomyocytes, in the presence and absence of serum ( $\pm$  serum). The conditioned media was changed 3 times, at intervals of 48 hours. Total RNA was isolated and RT-PCR was performed for expression of contractile proteins and cardiac specific genes. Immunostaining against myosin heavy chain,  $\beta$ -actin troponin-T and Troponin-I was performed.

**Results:** FACS analysis identified 66% of the hMSCs in the G1 phase. Differentiated hMSCs from the co-culture experiments expressed the mRNAs encoding myosin heavy chain,  $\beta$ -actin, and Troponin-T. Immunostaining was also positive against myosin heavy chain and troponin-T. In contrast, only  $\beta$ -actin expression was observed in the hMSC incubated with conditioned media  $\pm$  serum.

Conclusion: In addition to soluble signaling molecules, direct cell-to-cell contact is obligatory in relaying the external cues of the microenvironment controlling the differentiation of adult stem cells to cardiomyocytes. These data indicate that hMSCs are plastic and can be reprogrammed into a cardiomyogenic lineage that may be used in cell-based therapy for treating of heart failure.

## **Introduction:**

Cell transplantation is being explored as an alternative therapy for treating patients with end stage heart failure. Fetal cardiomyocytes (1,2), skeletal myoblasts (3,4,5), immortalized cell lines (6), fibroblasts (7), smooth muscle cells (8) and hematopoietic stem cells (9) have been transplanted into host myocardium. While this implantation was associated with improved cardiac function (10,11), evidence for normal electromechanical coupling between the implanted cells and host cardiomyocytes has been absent. All reported techniques of cellular cardiomyoplasty (CCM) have limitations and shortcomings, and these depend primarily upon the type of cell used for transplantation. Ideally, cells for use in CCM should be pluripotent, possess the capacity to differentiate to the desired cell type under appropriate stimuli, have a limited capacity to multiply, and should be capable of functional integration into the host myocardium.

The adult human mesenchymal stem cell (hMSC) has many of these characteristics, and as such may be highly suitable for CCM. Under appropriate stimuli these stem cells are highly plastic (12) and can differentiate into specialized tissues such as cartilage (13), osteocytes (14), adipocytes (15), chondroblasts (16), myogenic cells (17) and cardiac cells (18). Mesenchymal stem cells treated with azacytidine transdifferentiate into a cardiac phenotype *in vitro* (19). Moreover these stem cells can also differentiate into cardiomyocytes when injected into normal or acutely injured myocardium. However, the signals that are crucial for cardiac specific lineage are not well known. In order to improve the clinical utility of cell based therapy, the signaling pathways that induce the transformation of stem cells into cardiomyocytes need to be identified to help achieve successful engraftment.

Potential signals that direct stem cells to differentiate into cardiomyocytes include chemical (soluble) and mechanical (physical) factors. Stem cells differentiate into mature cells based on the signals from the microenvironment. The objective of our study was to determine if the signals that are produced by mature cardiomyocytes are sufficient to induce stem cells to differentiate into

cardiomyocytes, and if both physical contact and soluble factors are required for this process.

**Material and Methods:**

Three experimental groups of cell cultures were used to study the effects of soluble and mechanical factors on the transformation of stem cells into cardiomyocytes. In Group 1 (Figure 1a), adult human cardiomyocytes were cultured alone. In Group 2, (Figure 1b) hMSCs were cultured in media that had been conditioned by separate cultures of cardiomyocytes (“conditioned media”). In Group 3 (Figure 1a), stem cells and cardiomyocytes were cultured together (“co-culture”). In each group, cells were studied for the expression of cardiac-specific genes.

**Culture of human mesenchymal stem cells:**

Human mesenchymal stem cells were obtained from a commercial source (Biowhittaker Molecular Applications/Cambrex Inc, East Rutherford, NJ). The cells were originally isolated from the bone marrow of the posterior iliac crest of the pelvic bone of normal healthy volunteers and were positive for SH2, SH3, SH4, CD29 and CD44 and negative for CD14, CD 34 and CD45 as determined by flow cytometric analysis of surface antigens markers. These hMSCs were grown in human mesenchymal stem cell medium (HMSCM) containing 440 ml of basal medium, 50 ml of mesenchymal growth supplements (10% FBS), 200 mM of L-Glutamine, 25 units of penicillin and 25 µg of streptomycin at 37°C in a CO<sub>2</sub> incubator. At 80% confluence the hMSCs were split and subcultured.

**Culture of human cardiomyocytes:**

Cultured human cardiomyocytes were obtained from BioWhittaker Inc. The cardiomyocytes were grown in smooth muscle cell media containing 500 ml of smooth cell basal medium (SmBM) 0.5 mg/ml of hEGF, 5 mg/ml of insulin, 1 mg/ml of hFGF, 50 mg/ml gentamicin, 50 mg/ml of Amphotercin B and 5% fetal bovine serum.

**Preparation of conditioned media:**

The cardiomyocytes were cultured in T-25 cm<sup>2</sup> flask with 5 ml of Sm2 media for 48 hours. The resulting “conditioned media” was replaced with equal volume of fresh

media. Subsequently, the conditioned media was filtered using a 0.22  $\mu\text{m}$  filter and used to feed cultures of hMSCs.

#### **Fluorescent staining of the human cardiomyocytes:**

Prior to co-culturing cardiomyocytes with hMSCs, cardiomyocytes were labeled with green-fluorescent fluorescein diacetate (CMFDA, CellTracker, Molecular Probes Inc., Eugene OR) and sorted using fluorescent activated cell sorting (FACS). CMTDA is a fluorescent chloromethyl derivative that freely diffuses through the membranes of live cells. Once inside the cell, this mildly thiol-reactive probe undergoes a glutathione S-transferase-mediated reaction to produce a membrane-impermeant glutathione-fluorescent dye adduct. Briefly, CMTDA was mixed with prewarmed (37°C) serum free Sm2 media to a final concentration of 10  $\mu\text{M}$ . This concentration of probe was determined to be optimal for staining of cardiomyocytes using serial dilutions. The cardiomyocytes were incubated with the probe for 45 minutes at room temperature. The media was subsequently replaced with fresh serum free media and incubated for another 45 minutes to ensure complete modification of the probe and then the cells were washed with PBS to remove the excess fluorescent label.

#### **Co-culture of human cardiomyocytes and hMSCs:**

Labeled human cardiomyocytes and hMSCs were mixed at a ratio of 1:1 in Sm2 media, plated at a density of 10000 cells/cm<sup>2</sup> and incubated in a CO<sub>2</sub> incubator at 37°C for 48 hours. At the end of the experiment the cells were washed with PBS 3 times and 0.25% trypsin was added to detach the cells from the surface. Cardiomyocytes and treated hMSCs were separated using FACS prior to analysis.

#### **Culture of hMSCs with conditioned media:**

Media conditioned by cultures of human cardiomyocytes was filtered using a 0.22  $\mu\text{m}$  filter. Conditioned media (5 ml) was used to feed hMSC cultures for 48 hours, at which time the media was replaced with fresh conditioned media. The media was changed a total of 3 times at 48-hour intervals. Duplicate experiments were conducted



in which 10% serum was added to the conditioned media prior feeding the hMSCs. At the end of the experiment the cells were washed with PBS 3 times and 0.25% trypsin was added to detach the cells from the surface.

#### **Fluorescent activated cell sorting:**

After 48 hours of coculture, the differentiated hMSCs and the human cardiomyocytes were trypsinized and centrifuged at 700 rpm for 5 minutes. The cell pellet was suspended in 5 ml of PBS and the hMSCs were sorted and segregated at 488 nm optical filter. The hMSCs were collected and centrifuged at 500g for 10 minutes and processed later.

#### **Total RNA extraction, reverse transcriptase-PCR:**

Total RNA was extracted from untreated hMSCs (negative control), differentiated hMSCs and human cardiomyocytes (positive control) using an RNeasy Mini isolation kit (Qiagen Inc, Alameda CA). RT-PCR was performed to detect expression of myosin heavy chain,  $\beta$ -actin, troponin-T and Troponin-I using specific primers and Superscript One-Step RT-PCR (Invitrogen, Carlsbad CA) for cDNA synthesis. Pre-denaturation was performed at 50°C for 30 minutes and 94°C for 2 minutes. PCR amplification was carried out at 94°C for 30 sec, 60°C for 30 sec and 72°C for 30 sec for a total of 35 cycles and final extension of 1 cycle at 94°C for 7 minutes. The PCR products were size fractionated by 0.7% SeaKem GTG agarose gel electrophoresis. The following primers were used: Myosin heavy chain 5'-GGAGGAGGACAGGAAAAACCT-3' (forward), 5'-CGGCTTCAAGGAAAATTGC-3' (reverse), Troponin-T 5'-GGCAGCGGAAGAGGATGCTGAA-3' (forward), 5'-GAGGCACCAAGTTGGGCATGAACGA-3' (reverse),  $\beta$ -actin 5'-

CGCACCACTGGCATTGTCAT-3' (forward), 5'-TTCTCCTTGATGTCACGCAC-3' (reverse) and Troponin-I 5'-CCCTGCACCQGCCCAATCAGA-3' (forward), 5'-CGAAGCCCAGCCCGGTCAACT-3' (reverse) . Similarly, RT-PCR was performed for expression of the various connexins in the hMSC using specific primers as follows Cx-45 5'-CTATGCAATGCGCTGGAAACAACA-3' (forward) 5'-CCCTGATTTGCTACTGGCAGT -3' (reverse), Cx-40 5' ATGCACTGTGCGCATGCAGGA-3' (forward), 5'-CAGGTGGTAGTGTTTCCAGCCAG-3' (reverse), Cx-32 5'-CTGCTCTACCCGGGCTATGC-3' (forward), 5'-CTGCTCTACCCGGGCTATGC-3' (reverse), Cx-26 5'-CCGAAGTTCATGAAGGGAGAGAT-3' (forward), 5'-GGTCTTTTGGACTTCCCTGAGCA-3' (reverse), CX-43 5'-GAATTCTGGTTATCATCGTCGGGGAA-3' (forward), 5'-TACCATGCGACCAGTGGTGCGCT-3' (reverse)

### **Immunostaining:**

The differentiated hMSCs were sorted by FACS and cytopspined onto slides. The cells were fixed with acetone/methanol 50%/50% vol/vol for 5 minutes at room temperature and repeated twice. The slides were air dried overnight and bathed in 2 ml of PBS for 10 min. The slides were incubated with monoclonal IgG primary antibody for 45 minutes at room temperature. Primary antibodies were specific for  $\beta$ -Myosin Heavy Chain (MyHC) (A4.1025, 1:10 DHSB, University of Iowa), Sarcomeric Myosin (MF-20, 1:100 DHSB, University of Iowa), Cardiac Troponin-T (CT-3 1:100 DHSB, University of Iowa) and Troponin-I. The cells were washed with PBS and then incubated with secondary antibody (FITC-conjugated Affinipure goat anti-mouse IgG) diluted 1:200 for

45 minutes at room temperature. Fluorescence imaging was performed using Olympus AX-70 microscope and Open Labs Software.

## **Results:**

When initially plated, adult human mesenchymal stem cells appeared rounded in shape. After 24 hours after plating, the cells were adherent, elongated and spindle-shaped (Fig 2a). During mitosis the cells regained a rounded appearance and remained loosely attached until division was complete. At this phase of the cell cycle the cells flattened and elongated. The hMSCs were subcultured when they reach 70-80% confluence.

The cardiomyocytes had a rod-shaped morphology and were arranged in a syncytial fashion as shown (Fig 2b). Their particular phenotype lineage was established by immunostaining against myosin heavy chain and troponin-T. Human cardiomyocytes stained 100% with CMFDA

FACS analysis of the hMSC determined that 66% of the cells were in G1 phase. The remaining cells were in S phase (21%), and G2 phase (13%).

In the co-culture experiments, immunohistochemistry revealed an absence of staining for fast myosin (MF-1) in both cultured cardiomyocytes (Fig 3a) and transformed hMSCs (Fig 3b), and positive staining against sarcomeric myosin (MF-20),  $\beta$ -myosin heavy chain (MyHC), and troponin-T (CT-3) in cardiomyocytes (Fig 3c, e and g, respectively) and in transformed hMSCs (Fig 3d, f and h, respectively). There was no staining against Troponin-I protein at 2 days of co-culture (Fig.3h). In addition, RT-PCR revealed the expression of myosin heavy chain,  $\beta$ -actin and cardiac troponin-T in cell-to-cell contact co-culture (Fig 4). Untreated hMSCs and human cardiomyocytes were used as negative and positive controls, respectively. In the conditioned media experiments,  $\beta$ -actin expression was noted in the hMSCs exposed to cardiomyocyte-conditioned media, both in the presence and absence of serum. There was no expression of the  $\beta$ -myosin heavy chain, troponin-I or troponin-T in hMSCs treated with the conditioned media (Table I).

Importantly, the expression of gap junction proteins (Cx 40, Cx-43, Cx-45, Cx-32) was identified in untreated cultured hMSCs. This finding is particularly encouraging since gap-junctions are critical to establishment of cell-to-cell electrochemical coupling (Fig 5).

**Discussion:**

These studies demonstrate that adult human mesenchymal stem cells have the potential to differentiate into cardiomyocytes under the appropriate microenvironment. Under co-culture conditions, when there was direct contact between cardiomyocytes and hMSCs, the hMSCs begin to express the cardiac-specific proteins myosin heavy chain, beta-actin and troponin-T. Most importantly, only  $\beta$ -actin was expressed when the hMSCs were cultured with conditioned media, either in the presence or absence of serum, when there was no physical contact between the cardiomyocytes and hMSCs.

Elements of the microenvironment provide the critical signals to direct and control differentiation of human mesenchymal stem cells to a cardiac lineage. The potential factors involved are numerous, but may be broadly characterized as either chemical (soluble) or physical (mechanical). In these experiments, we demonstrated that the soluble factors alone were not sufficient to induce differentiation of hMSCs into cardiomyocytes, and that physical contact between the cardiomyocytes and hMSCs is necessary under these conditions. However, the cell density may be critically important since the expression of cardiac specific proteins occurred when cardiomyocytes and hMSCs were co-cultured at 1:1 ratio, but were not seen at other plating ratios in preliminary work performed in our lab.

During embryogenesis, stem cells differentiate to form the cells that comprise the developing organs and tissues. In the adult, some tissues are able to regenerate lost or damaged cells through the differentiation of progenitor cells or additional stem cells. However, cardiomyocytes are not readily replaced with contractile cells when they are lost as a result of myocardial infarction. Cardiomyocytes are in a dormant phase of the cardiac cycle, and do not undergo cellular division. However, the hMSCs progress through the cell cycle and are capable of cellular division. The observation that 66% of the hMSCs were in G1 phase of the cell cycle suggests that prolongation of this phase could play an

important role in directing hMSC commitment. Since cardiomyocytes are incapable of division, and no readily available source of replacement cells exist, lost cells are partially replaced with scar tissue, while neighboring cardiomyocytes hypertrophy in an effort to restore cardiac function. There is no clear understanding why stem cells that are capable of producing cardiomyocytes in the embryo are incapable of such a task in the adult heart. Recent evidence demonstrates that such replacement may occur to a small degree, but that the rate may be too slow to be functionally significant (20). The purpose of these experiments was to try to elucidate some of the signals that are involved in the differentiation of mesenchymal stem cells into cardiomyocytes. Through a clear understanding of these processes, we may be able to manipulate stem cells such that they may be the ideal source of cells for cellular cardiomyoplasty, or that we may be able to direct them migrate to the myocardium to replenish cardiomyocytes at a rate that can be clinically useful..

Previous work has demonstrated the ability of stem cells to undergo differentiation into cardiomyocytes, but these studies have not examined the mechanisms of the differentiation process. It is clear that the microenvironment of the cells is an important component of this process. If stem cells are implanted into myocardial scar, they differentiate into a variety of non-myocyte cell types (21), reflecting the prevailing influence of the myocardial microenvironment. Conversely, if stem cells are implanted into an “optimally” permissive microenvironment, they may selectively differentiate into cardiomyocytes. Stem cells may also be used to repopulate the heart without direct injection into the myocardium. In mice that have undergone bone marrow transplantation and subsequent regional myocardial infarction, marrow-derived cardiomyocytes and endothelial cells have been found in the peri-infarct region (9). Finally, host-derived cardiomyocytes have been located in small numbers in transplanted human hearts that have been later excised (22). These data demonstrate both the ability of adult mesenchymal stem cells to differentiate into cardiomyocytes under the proper conditions, and also reflect the pluripotent nature of these stem cells in that they also can form the supporting structures of the heart. Finally,

stem cell-derived cardiomyocytes have only been located in hearts subjected to injury, and not under normal conditions. This suggests that the microenvironment of the stem cell is critical in permitting the process differentiation, and that the stem cells are responding to some factor(s), either chemical or mechanical, are active during these periods of stress or injury.

The processes that regulate cell differentiation are complex, and the interactions between signals are largely unknown. However, it is clear that the microenvironment of the developing cell plays a critical role in determining its ultimate fate. The components of the microenvironment that influence cellular differentiation can be broadly classified as either chemical or physical (mechanical). The chemical signals include cytokines, hormones, ionic gradients, and other soluble factors that are produced by either neighboring or distant cells. The mechanical factors can be equally complex, and may include stimulation of receptors through direct contact with neighboring cells or by the components of the extracellular matrix (ECM), the influences of cell stretch or other forces, the electrical environment of the cells, and perhaps even other signals.

Co-culture and conditioned media techniques provide an excellent model to study the signals that influence cellular development and differentiation. Through the use of co-culture techniques, stem cells are exposed to many of the physical and chemical signals that are present within the native myocardium, particularly the mechanical signals that are produced through direct cell-to-cell contact between the cardiomyocytes and hMSCs. In contrast, the experiments that use conditioned media provide the soluble factors that are elaborated from the cardiomyocytes without allowing direct cell-to-cell contact, thus separating the effects of the chemical stimuli from the physical. While the conditioned media contains cytokines and other soluble elements that are elaborated from the cardiomyocytes during normal growth, these are not sufficient to stimulate the stem cells to differentiate into cardiomyocytes. In the conditioned media experiments, the signals that are related to direct contact between the cell types are lacking. This suggests that the cell-to-cell contact that is possible between

cardiomyocytes and hMSCs in the co-culture scenario is critical in the differentiation process.

Cell-cell interaction is a complex phenomenon, which involves contact between cells through junctional complexes including tight junctions, desmosomes and gap junctions. For this reason, it is important that the transformed stem cells are capable of expressing the connexins that are critical in the formation of gap junctions. The expression of the gap junction proteins seen in the untreated hMSCs is critical for the complete expression of cardiac phenotype and functional integration into the host tissue. These proteins may be involved in the signal transduction that occurs between the hMSCs and cardiomyocytes during the co-culture experiments. Perhaps the connexins have been arranged into dormant channels that open with cell-to-cell contact and aid in the transfer of intracellular signaling molecules.

Cell contact can also result in changes in cell shape due to mechanical stretch imposed by neighboring cells. In addition, homology of cell surface receptors and other proteins involved in cell-cell adhesion between the cardiomyocytes and stem cells could activate the differentiation-associated genes and alter the genotype and phenotype. The intracellular signal transduction pathways may be triggered by transmembrane receptors such as epidermal growth factor; platelet-derived growth factor via autophosphorylation and by binding with ligands that regulate the transmission of MAP kinase signaling pathway or activation of protein kinase C pathway via hydrolysis of phosphoinositol.

Evaluation of the factors required to promote stem cell differentiation is critical in refining the techniques of cellular cardiomyoplasty. Current techniques are inadequate to produce a clinically significant increase in cardiac function for a variety of reasons. When skeletal myoblasts or other non-cardiomyocytes are used, there is not complete electromechanical integration of the implanted cells with the native cardiomyocytes. While there may be a concomitant improvement in cardiac function, this is likely due to changes in the diastolic properties of the ventricle. As such, maximal benefit can only be obtained if the transplanted cells



have the capacity to differentiate into cardiomyocytes that can fully integrate into the myocardium. Another concern with the current techniques of CCM is the low rate of integration of implanted cells. Only a small fraction of the implanted stem cells remain viable in the myocardium after CCM, limiting the therapeutic benefit of this technique using current methods.

If the signals that are involved in the transformation of stem cells into cardiomyocytes can be understood, then the rate of functional integration of these cells, and thus the success of CCM, can be improved significantly. Since these results suggest that mechanical factors are important in the transformation process, then it is possible that stem cells can be pretreated in vitro in an environment that mimics many of these conditions prior to implantation. By doing this, it may be possible to commit the stem cells to a cardiomyocytes lineage prior to implantation to increase the therapeutic yield of CCM.

#### **Limitations of This Study:**

These studies were conducted using an in vitro controlled microenvironment and observed for relatively short period of time (48 hours). While several myogenic markers were expressed in the stem cells subjected to co-culture,  $\beta$ -myosin heavy chain,  $\beta$ -actin, troponin-T are also expressed to varying degrees by skeletal muscle cells. In addition, troponin-I was not seen in the treated stem cells. However, the time frame of the study was relatively short, and it may take upwards of seven days before troponin-I expression may be detectable in transforming cells. Future studies will need to look at this phenomenon a longer time points in order to see the expression of other cardiac proteins. In addition, it is anticipated that electromechanical integration of the hMSCs with the cultured cardiomyocytes will also occur as a late finding, if the cell cultures can be maintained long enough.

Cell fusion (23) between the cardiomyocytes and hMSCs remains a concern when interpreting these observations and those of related studies of gene expression in differentiating stem cells. The fact that Troponin-I expression was selectively absent from co-cultured hMSCs demonstrates that fusion is not

responsible for the findings. A related issue is the chance that separation of the cardiomyocytes and hMSCs was not complete, and that RT-PCR amplified genes from the cardiomyocytes that contaminated the hMSC cell population. If this were to have occurred, then the RNA encoding troponin-I would have been seen in the stem cells subjected to co-culture.

**Conclusion:**

In addition to soluble signaling molecules, direct cell-to-cell contact is obligatory in relaying the external cues of the microenvironment controlling the differentiation of adult stem cells to cardiomyocytes. These data indicate that hMSCs are plastic and can be reprogrammed into a cardiomyogenic lineage that may be used in cell-based therapy for treating of heart failure.

## References:

1. Soonpaa MH, Koh GY, Klug MG, Field LJ. Formation of nascent intercalated discs between the grafted fetal cardiomyocytes and host myocardium. *Science* 1994; 264:98-101.
2. Reinecke H, Zhang M, Bartosek T, Murry CE. Survival, integration, and differentiation of cardiomyocyte grafts: a study in normal and injured rat hearts. *Circulation* 1999; 100(2): 193-202.
3. Chiu RCJ, Zibaitis A, Kao RL. Cellular Cardiomyoplasty: myocardial regeneration with satellite cell implantation. *Ann Thorac Surg* 1995; 60 (1):12-15
4. Atkins BZ, Lewis CW, Kraus WE, Hutcheson KA, Glower DD, Taylor DA. Intracardiac transplantation of skeletal myoblasts yields two populations of striated cells in situ. *Annals of Thoracic Surgery*. 1999; 67(1): 124-9.
5. Taylor DA. Atkins BZ. Hungspreugs P. Jones TR. Reedy MC. Hutcheson KA. Glower DD. Kraus WE. Regenerating functional myocardium: improved performance after skeletal myoblast transplantation [published erratum appears in *Nat Med* 1998; 4(10): 1200]. *Nature Medicine*. 1998; 4(8): 929-33.
6. Koh GY. Soonpaa MH. Klug MG. Field LJ. Long-term survival of AT-1 cardiomyocyte grafts in syngeneic myocardium. *American Journal of Physiology*. 1993; 264(5 Pt 2): H1727-33.
7. Sakai T, Li RK, Weisel RD, Mickle DA, Kim EJ, Tomita S, Jia ZQ, Yau TM. Autologous heart cell transplantation improves cardiac function after myocardial injury. *Ann Thorac Surg*. 1999; 68(6): 2074-80; discussion 2080-1.
8. Yoo KJ, Li RK, Weisel RD, Mickle DA, Li G, Yau TM. Autologous smooth muscle cell transplantation improved heart function in dilated cardiomyopathy. *Ann Thorac Surg*. 2000; 70(3): 859-65.
9. Jackson KA, Majka SM, Wang H, Pocius J, Hartley CJ, Majesky MW, Entman ML, Michael LH, Hirschi KK, Goodell MA. Regeneration of ischemic cardiac muscle and vascular endothelium by adult stem cells. *J Clin Invest*. 2001 Jun;107(11):1395-402.

10. Li RK, Mickle DAG, Weisel RD, Zhang J, Mohabeer MK. In vivo survival and function of transplanted rat cardiomyocytes. *Circ Res* 1996; 78: 283-288
11. Scorsin M, Hagege AA, Dolizy I, Marotte F, Mirochnik N, Copin H, Barnoux M, le Bert M, Samuel JL, Rappaport L, Menasche P. Can cellular transplantation improves function in doxorubicin-induced heart failure? *Circulation*. 1998; 98 (19 Suppl):II151-5; discussion II-155-6.
12. Pittenger MF, Mackay AM, Beck SC, Jaiswal RK, Robin Douglas, Mosca JD, Moorman MA, Simonetti DW, Stewart Craig, Marshak DR. Multilineage potential of adult human mesenchymal stem cells. *Science* 1999; 284:143-147
13. Ashton B.A, Allen TD, Howlett CR, Eaglesom CL, Hattori A, Owen M. Formation of bone and cartilage by marrow stromal cells in diffusion chambers in vivo. *Clinical Orthopedics*. 1980; 151:294-307.
14. Richard, DJ, Sullivan, TA, Shenker, BJ, Leboy PS, Kazhdan I. Induction of rapid osteoblast differentiation in rat bone marrow stromal cells cultures by dexamethasone and BMP-2. *Dev Bio* 1994; 161: 218-228
15. Umezawa A, Maruyama T, Segawak K, Shadduck RK, Waheed A, Hata J. Multipotent marrow stromal cell line is able to induce hematopoiesis in vivo. *J Cell. Physiol* 1992; 151: 197-205
16. Howlett CR, Cave J, Williamson M, Farmer J, Ali SY, Owen ME. Mineralisation in vitro cultures of rabbit marrow stromal cells. *Clinical orthopaedic Related Research* 1986; (213): 251-263
17. Ferrari G, Angelis GC, Colletta M, Paolucci E, Stornaiuolo A, Cossu G, Mavilio F. Muscle regeneration by bone marrow derived myogenic progenitors. *Science* 1998; 279: 1528-1530
18. Toma C, Pittenger MF, Cahill KS, Byrne BJ, Kessler PD. Human mesenchymal stem cells differentiate to a cardiomyocyte phenotype in the adult murine heart. *Circulation*. 2002 Jan 1;105(1):93-8.
19. Makino S, Fukuda K, Miyoshi S, Konishi F, Kodama H, Pan J, Sano M, Takahashi T, Hori S, Abe H, Hata J, Umezawa A, Ogawa S. Cardiomyocytes

- can be generated from marrow stromal cells in vitro. *Journal of Clinical Investigation*. 1999; 103(5): 697-705.
20. Beltrami AP, Urbanek K, Kajstura J, Yan SM, Finato N, Bussani R, Nadal-Ginard B, Silvestri F, Leri A, Beltrami CA, Anversa P. Evidence that human cardiac myocytes divide after myocardial infarction. *N Engl J Med*. 2001 Jun 7;344(23):1750-7.
21. Wang JS, Shum-Tim D, Chedrawy E, Chiu RC. The coronary delivery of marrow stromal cells for myocardial regeneration: pathophysiologic and therapeutic implications. *J Thorac Cardiovasc Surg*. 2001 Oct;122(4):699-705
22. Laflamme MA, Myerson D, Saffitz JE, Murry CE. Evidence for cardiomyocyte repopulation by extracardiac progenitors in transplanted human hearts. Evidence for cardiomyocyte repopulation by extracardiac progenitors in transplanted human hearts. *Circ Res*. 2002 Apr 5;90(6):634-40.
23. Terada N, Hamazaki T, Oka M, Hoki M, Mastalerz DM, Nakano Y, Meyer EM, Morel L, Petersen BE, Scott EW. Bone marrow cells adopt the phenotype of other cells by spontaneous cell fusion. *Nature*. 2002 Apr 4;416(6880):542-5.

Figure 1a

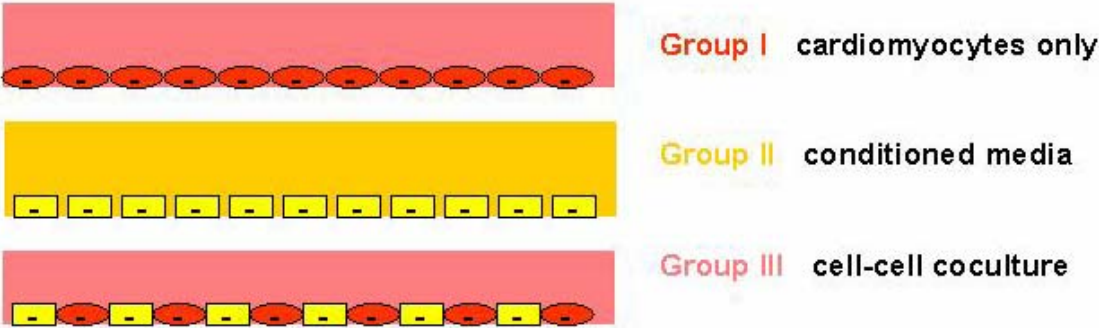


Figure 1b:

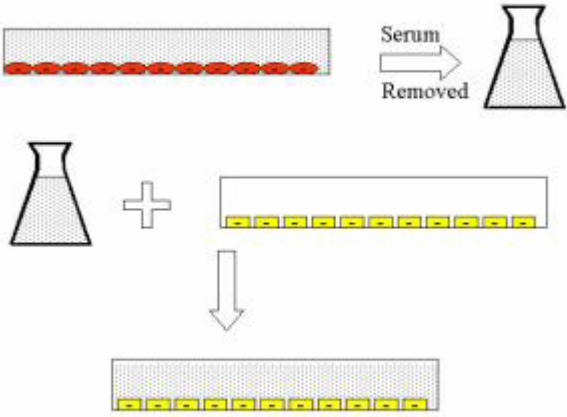


Figure 2:

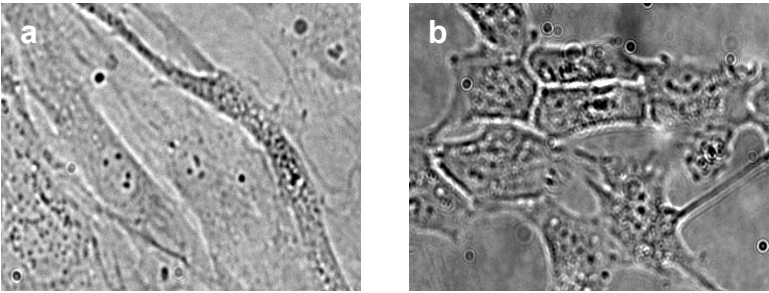


Figure 3:

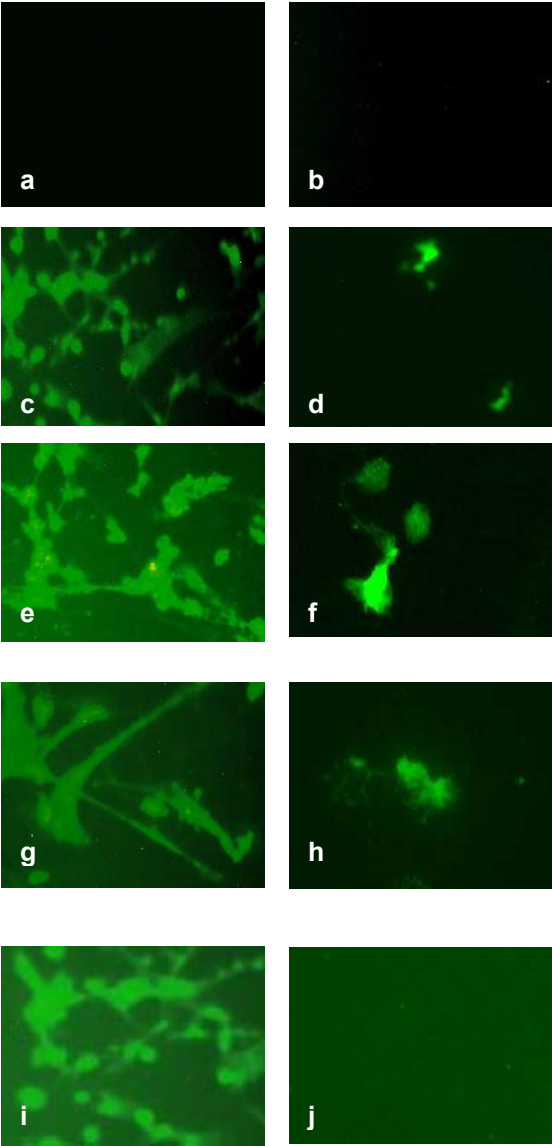


Figure 4

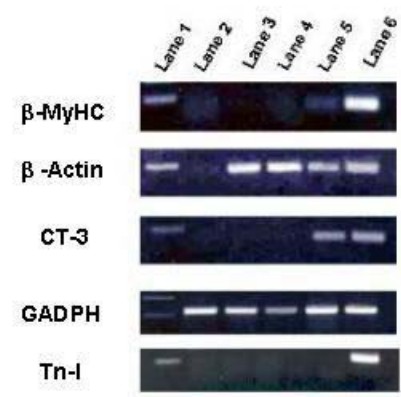




Figure 5

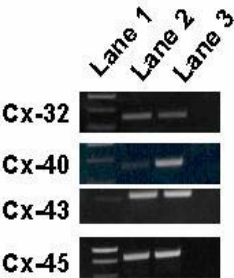


Table I:

	$\beta$ - MyHHC	$\beta$ -Actin	Troponin-T	Troponin-I
<b>Group I</b>	+	+	+	+
<b>Group II</b>	–	+	–	–
<b>Group III</b>	+	+	+	–

Table II:

	hMSC	Cardio-myocyte
Cx-26	–	–
Cx-32	+	+
Cx-40	+	+
Cx-43	+	+
Cx-45	+	+

Figure 1:

Figure 1a:

Group I cardiomyocytes grown in the tissue culture plates

Group II Conditioned media prepared from the cardiomyocytes after 48 hours in culture

Group III Coculture of hMSC with cardiomyocytes.

Figure 1b:

Conditioned media prepared from the soluble factors released from the cardiomyocytes, in which hMSC were subsequently cultured.

Figure 2:

Figure 2a Spindle shaped human mesenchymal stem cells (20X) 24 hours after seeding on cell polystyrene cell culture surface.

Figure 2b Rod shaped human cardiomyocytes forming intercellular junctions with neighboring cells.

Figure 3: (a-h) Left panel are positive controls, consisting of cardiomyocytes. Right panel are transformed hMSCs. (10X)

a: No staining observed against fast myosin (MF-1) in human cardiomyocytes

b: No staining observed against fast myosin (MF-1) in untreated hMSCs (negative control)

d: Positive staining for sarcomeric myosin (MF-20) in co-cultured hMSCs

f: Positive staining for heavy chain myosin ( $\beta$ -MyHC) in co-cultured hMSCs

h: Positive staining for troponin-T in co-cultured hMSCs.

j: Negative staining against troponin-I

c,e,g,i: Positive staining in cultured cardiomyocytes for MF-20,  $\beta$ -MyHC, troponin-T and troponin-I respectively.

Figure 4: RT-PCR for specific hMSC transformation markers (48 hr-coculture). Lane 1: 1kb DNA Ladder, Lane 2: hMSC (negative control), Lane 3: conditioned media treated hMSC in the presence of serum, Lane 4 conditioned media treated with hMSC in the absence of serum, Lane 5: cell-cell contact dependent coculture. Lane 6: human cardiomyocytes (positive control).

$\beta$ -MHC,  $\beta$ -actin and CT-3 expression in cell-cell contact dependent coculture. Expression of  $\beta$ -actin in conditioned media  $\pm$  serum. GAPDH is internal control. Tn-I is troponin-I expression in cultured cardiomyocytes.

Figure 5: Expression of gap junctions connexin specific genes (Cx-32, Cx-40, Cx-43 and Cx-45) in hMSC. Lane 1: 1kb DNA Ladder, Lane 2: hMSC expression of specific connexins, Lane 3: Expression of connexin specific genes in human cardiomyocytes (positive control)

Table 1: Group I (cardiomyocytes alone), showing the expression of  $\beta$ -myosin heavy chain,  $\beta$ -actin, troponin-T and troponin-I. Group II (hMSC exposed to conditioned media) show only expression of  $\beta$ -actin. Group III (cell-cell contact co-culture) shows expression of  $\beta$ -myosin heavy chain,  $\beta$ -actin and troponin-T.

Table II: Expression of Connexin specific genes in untreated hMSC and adult cardiomyocytes.

# REMOTE MONITORING OF CELLULAR NETWORK ASSEMBLY AND FUNCTION

Vijay Noronha<sup>2</sup>, Can Evren Yarman<sup>2</sup>, J. Yasha Kresh<sup>1</sup>, Banu Onaral<sup>2</sup>

<sup>1</sup>Depts. of Cardiothoracic Surgery / Medicine, Medical College of Pennsylvania/Hahnemann University, PA, USA

<sup>2</sup>School of Biomedical Engineering, Science and Health Systems, Drexel University, PA, USA

**Abstract-** A video microscopy based collaboratory has been developed to study cellular network dynamics, in particular, to monitor live-cell spatio-temporal organization in real-time. The aim is to investigate the effects of intercellular communication on tissue genesis, differentiation and cell survival.

The platform enables multiple researchers to remotely access a digital microscopy system consisting of very high-resolution CCD-imaging technology coupled with real-time (~1Gbps) data transfer throughput. Remote control and image acquisition facilitates collaboration between cellular biologists, tissue engineers and computational scientists studying complex cellular organization dynamics and assembly.

The remote control microscope is connected to a local 'Server' whose software is written in Java 2 (jdk 1.3) and can be accessed by a 'Client' using any web browser supporting Java. Any Internet user can control the microscope and interact with other users who are on-line or are directly connected. This interactive environment does not impose any hardware or software limitations on the 'Clients'.

A remote user can control the movement of the stage in X-Y axes, control focus (Z axis), change magnification, change excitation and light emission filters and their corresponding shutters, acquire topographic and fluorescent images from the microscope and process these images.

The tele-microscopy based environment serves as a focal point for investigators with different expertise to collaborate synergistically on projects that require multi-disciplinary approaches.

**Keywords -** Telemicroscopy, Collaboratory, Biocomplexity, Live-Cell Observatory

## I. INTRODUCTION

The 'Cellular Network Dynamics' project aims to monitor cellular networks in real-time in order to gain insight into their biological organization. The specific aims are: (i) to study the live-cell spatio-temporal organization, the role of intercellular communication and its effects on cell culture 'function' and 'survival'; (ii) to develop computational models of cellular network signaling and adaptability. A microscope-based live video system (cellular observatory) is implemented to enable tissue visualization, image analysis and cellular network modeling.

This paper describes the implementation of a distributed high-resolution digital microscopy system for visualizing cellular network assembly and function. The "Telemic" remote-control microscopy system [1,2] enables a shared (client-server mode) observatory (Fig. 1) to study the evolution and communication in cellular networks.

## Interaction and Pattern Formation in Cellular Networks

Networks are ubiquitous in the biological world. Multicellular organisms develop by transforming a collection of undifferentiated cells into intricately organized groups of cells that coalesce to form functional structures. This process known as pattern formation is responsible for structural differentiation, including rhythm formation in cardiac tissue. The ability to dynamically detect cellular and sub-cellular network dependent events is a first step in developing models of cellular organization. Video monitoring of cellular activity in tissue culture using immuno-fluorescence localization of inter- and intracellular integrity (gap junctions, integrins, actin fibers) makes possible studies of cellular network dynamics (signaling complexity, self-organization). As the numbers of attachment points (nodes) in these cellular networks increase and the connectivity becomes more efficient, one would expect the emergence of global, self-organized, coherent tissue structures with new and complex phenotype expression (i.e., network adaptation and stress tolerance). The common problem in many highly coupled systems is how the network structure (e.g. neurons; societies and Internet) facilitates and constrains its own behavior and/or mediates the propagation of stress/failure [3].

## Cellular Observatory

Existing methods monitor singular cellular / sub-cellular events and or produce only a static 'snap-shot' of a specific moment of a dynamic process of interest. Conventional 'fixed-cell' assays are inherently deficient in elaborating the dynamic links between the inter- and intracellular processes that define cellular homeostasis. In contrast, simultaneous tracking of a set (e.g. three or more) of cell specific functions in each of the 'units' of the cellular assembly can help unravel some of the critical links that are involved in the dynamic re-organization of cellular network behavior.

The specific methodological approach under development in our laboratory monitors cellular organization by binding fluorescent markers to specific structures within the living cell and recording real-time images of cellular culture activity, spanning many hours of observation. The use of multiple fluorescent markers help inter-relate cell function (e.g. cytoskeletal meshwork, gap junction distribution) to the functional inter-connectivity of the cellular network, as well to its viability. The acquired images form the basis for the time-lapse mapping of processes and modeling of the emergent intercellular communication patterns. The sequential frames (pattern formation) are used as a baseline to study the role of spatio-temporal organization in the

adaptation of the formed cellular network to the imposed environmental changes.

The distinguishing features of the live-cell telemicroscopy system is its modularity and universal applicability, and in particular, its ability to remotely acquire and control fluorescent and brightfield images in real-time, spanning hours to days of observation.

In addition, the concept of a distributed 'collaboratory' i.e., a multidisciplinary environment without walls ("virtual center"), assembled and constituted by the research areas and expertise of its members is exemplified by this initiative. A major rationale advanced for this approach is that "It is easier to move information around than the requisite technology or people."

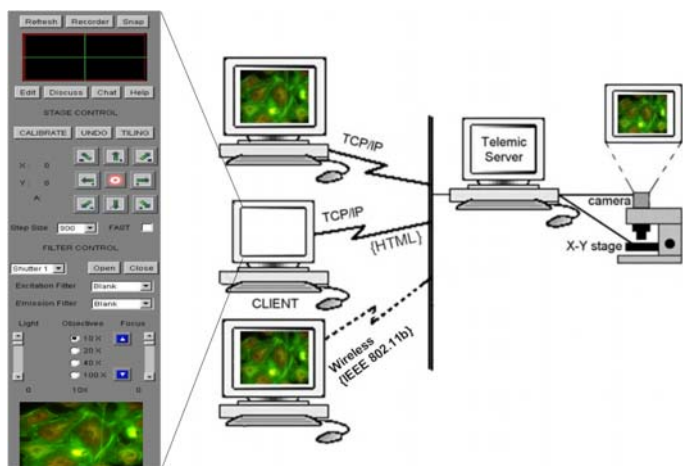


Fig 1. Telemicroscopy (Client-Server Architecture)

## II. METHODOLOGY

The software for remote control microscopy was initially developed utilizing a very high-speed optical link with 1 Gbps IP infrastructure (gigabit per second Ethernet + 2.5 Gbps SONET), ultra-high capacity router and ultra-high speed non-linear optical regeneration, between Drexel main campus and MCP-Hahnemann University for controlling the Telemic microscope and associate components:

- 1) Microscope (Nikon Diaphot 300): The Diaphot 300 is a research level inverted optical microscope with multiple optical output ports and epi-fluorescence capability
- 2) Digital Controller (Ludl MAC 5000 with RS232 and USB communication): This is a high performance modular control system designed to automate and remote control the microscopy functions (X-Y and Z movement, Filter and shutter control). This module automates the control of:

a) <X-Y Movements>: This module controls the movement of the stage in the X-Y plane and has the following features:

- Incremental (0.2 micron / step) movements in the any X-Y directions.
- Ability to return back to the set origin from any given position
- Calibrate the stage by prompting its movement to the respective X-Y limits.
- Ability to move the stage to any specified X-Y coordinates, within the available field of excursion. This functionality is graphically emulated on the screen that maps the area available for stage movement and then provides cursors to mark the x-y coordinates.
- Ability to undo stage movements.
- Ability to control the speed of the stepper motors controllable by the 'Client' -the stage can be directed to move at a prescribed speed.

b) <Focus Control> (Z axis): The focus control system provides reproducible, high-resolution automated control of the microscope focus using a micro-stepping motor 1 (0.02 micron/step -step size.)

c) <Filters Wheels>: This module allows the changing of filters at Excitation (illumination end) and at the Emission (camera end) with 30 msec between switched positions.

d) <Shutter Control>: This module controls the opening and closing of shutters of the excitation and emission filter wheels. Closing the shutters between image acquisitions prevents light bleaching (e.g. fluorescent marker) of cell culture during prolonged observation (time-lapsed motion studies).

### 3) Digital Camera:

**Camera 1) Roper CoolSnapFX** (12-bit dynamic range @ 1300 x 1030) A cooled (-30°C) digital CCD camera designed for fluorescence imaging

**Camera 2) Sony DFW-V300** (30 fps @ (640 x 480) VGA) is a digital video camera which uses the "IEEE-1394" (Firewire) computer interface.

The remote control microscope is connected to a local 'Server', which can be accessed by a 'Client' using any web browser that supports Java. The 'Server' software is written in Java 2 (jdk 1.3). Any Internet user can control the microscope and interact with other users who are on-line or are directly connected. This interactive environment does not impose any hardware or software limitations on the 'Clients'.

The Telemicroscopy 'Server' runs on a standard Windows NT workstation with a fixed IP address. The 'Server'

automates functions such as image acquisition and stage, focus, filter and shutter control.

The image acquisition module interfaces with the camera and captures images at a specified rate (e.g. 10 fps) and writes them to disk in jpeg (compressed) format.

The 'Server' also provides a <Chat> function, which allows on-line users to converse with other Telemic subscribers.

A <Discuss> mode was implemented to enhance the collaboration functionality to enable users to exchange information about a captured image among each other. In addition, a dynamic pointer is invoked such that it can be "surrendered" to any one of the users in the <Discuss> Mode. If a user moves the arrow around and repositions it, then a corresponding arrow on all of the active screens will be repositioned accordingly.

The Telemicroscopy 'Client' is a Java applet (Fig. 2) that emulates the functions of a microscope and can be viewed in any java-enabled browser. The java 'Client' provides an interface to facilitate the following functions – <X-Y stage movement control> <filter wheel> and <shutter control>, <focus control>, <image update>, <image snapping>, <image editing> (filtering), <chat client>, <discuss client>, <video recorder> to display images at user-specified rates from the 'Server', <illumination control> and <changing objectives>.

The 'Client' provides an <EDIT> feature, which allows editing of captured images. Image processing functions currently available are <Grayscale>, <Invert>, <Blur>, <Sharpen> and <Contrast>. This module can be expanded to include various nonlinear filtering algorithms and user-specific image processing routines such as classification counting, size distribution and analysis

The Java 'Client' updates its image after every successful 'Server' command execution. The "virtual" <Video Recorder> feature allows users to view real time video at user specified frame rates. (Default-10 fps)

The <Tiling> feature allows a user to create a montage (topographic map) of a specific region of the cell culture. This allows automatic control of the stage to facilitate capturing sequential "tiles" of the desired area and then reassembling them into a composite representation.

To observe live cell phenomenon (e.g. making and breaking of communication channels) brightfield imaging is combined with fluorescence imaging. Brightfield imaging using a digital video camera (Sony DFW-V300- 30 fps) camera provides a topographic view of the specimen under observation. Fluorescence imaging enables monitoring of cellular organization by binding fluorescent markers (GFP) to specific intra- or inter-cellular structures and recording the real-time images of cellular network activity, spanning many hours of observation to make time-lapsed movies. The use of fluorescent markers helps relate specific cell phenomenon to the network emergent function. The live-cell imaging system combined with a regulated micro-environment (temperature, media composition, atmospheric gases) control is a basic

requirement for long-term observation of the dynamic organization of cellular networks.

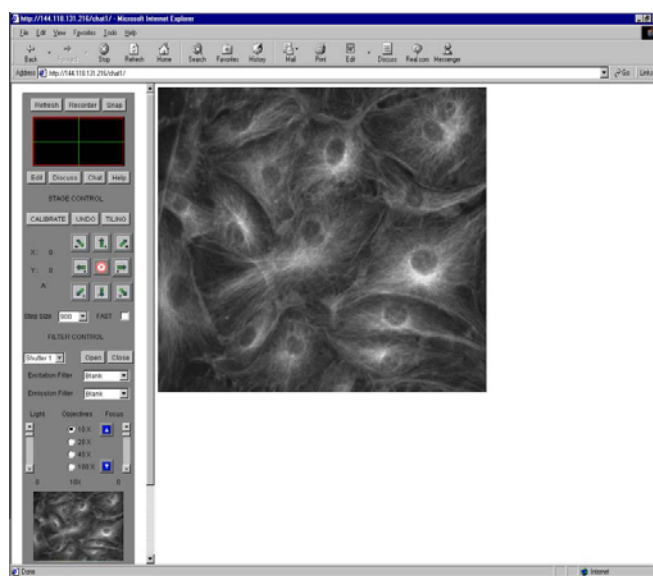


Fig 2. Telemicroscopy Applet viewed by Microsoft Internet Explorer (Left - The microscope control and image acquisition interface. Center - updated image from the microscope)

### III. ABBREVIATIONS AND ACRONYMS

- 1) Telemicroscopy: Remote Control Live Cell Virtual Observation Platform
- 2) Telemic: Refers to Telemicroscopy 'Client' or 'Server' or component module
- 3) Biocomplexity: Study of complex systems occurring within organisms
- 4) GFP- Green Fluorescent Protein

### IV. CONCLUSION

The imaging platform presented in this paper is being used in studies of cellular network dynamics, in particular, the monitoring of live-cell spatio-temporal organization and the role of intercellular communication and its effects on tissue genesis, differentiation and survival.

The distinguishing features of the live-cell telemicroscopy system are 1) its modularity and universal applicability, 2) ability to remotely acquire and control fluorescent and brightfield images in real-time spanning hours to days of observation.

The digital microscopy collaboratory environment serves as a focal point for investigators with different expertise to collaborate dynamically on projects that require multidisciplinary approaches.

The long-term objective of this research is to reverse-engineer biological principles and help inspire the design and engineering of robust and adaptive communication networks and intelligent systems. The advantages gained in mimicking biological organization and function are particularly intriguing. The cellular network models may have applicability in areas such as emergent communication networks, evolutionary/adaptive optimization of networks, and distributed memory/"smart agents". In addition, the "biocomplexity" inspired concepts of self-regulation [3] and assembly can be explored "in-silico" and generalized to study autonomous agent models; 2-D cellular automata; "flocking" self-organized/coalition behavior; search pattern evolution problems; evolutionary optimization problem; spatial genetic algorithms among others.

## V. ACKNOWLEDGMENT

This effort is supported in part by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory, Air Force Material Command, USAF.

## VI. REFERENCES

- [1] Wolf G, Petersen D, Dietel M, Petersen I: *Telemicroscopy via the Internet*, Nature, 391:613, 1998.
- [2] Onaral, B., Kresh, J.Y., Luzuria, E. "Complex Biological Systems" Plenary Lecture: IEEE-EMBS Asia-Pacific Conference on Biomedical Engineering, September 26-28, 2000, China
- [3] Kresh, J.Y., Izrailtyan, I. "The Heart as a Complex Adaptive System". In: *Unifying Themes in Complex Systems*, Y. Bar-Yam, Ed., Perseus Books, 2001.



# A novel fast optical switch based on two cascaded Terahertz Optical Asymmetric Demultiplexers (TOAD)

Bing C. Wang, Varghese Baby, Wilson Tong, Lei Xu, Michelle Friedman, Robert J. Runser, Ivan Glesk, Paul R. Prucnal

*Dept. of Electrical Engineering, Princeton University  
Princeton, NJ, USA 08544  
bingwang@princeton.edu*

**Abstract:** A novel optical switch based on cascading two terahertz optical asymmetric demultiplexers (TOAD) is presented. By utilizing the sharp edge of the asymmetric TOAD switching window profile, two TOAD switching windows are overlapped to produce a narrower aggregate switching window, not limited by the pulse propagation time in the SOA of the TOAD. Simulations of the cascaded TOAD switching window show relatively constant window amplitude for different window sizes. Experimental results on cascading two TOADs, each with a switching window of 8ps, but with the SOA on opposite sides of the fiber loop, show a minimum switching window of 2.7ps.

© 2002 Optical Society of America

**OCIS codes:** (060.0060) Fiber optics and optical communications; (230.1150) All-optical devices

---

## References and links:

1. M. W. Chbat, B. J. Hong, M. N. Islam, C. E. Soccolich, P. R. Prucnal "Ultrafast Soliton Trapping AND Gate," *J. Lightwave Technol.* **10**, 2011-2016 (1992)
2. N. J. Doran and D. Wood "Non-linear optical loop mirror," *Opt. Lett.* **14**, 56-58 (1988)
3. J. P. Sokoloff, P. R. Prucnal, I. Glesk, and M. Kane "a Terahertz optical asymmetric de-multiplexer (TOAD)," *IEEE Photon. Technol. Lett.* **5**, 787-790 (1993)
4. M. Eiselt "Optical loop mirror with semiconductor laser amplifier," *Electron. Lett.* **28**, 1505-1507 (1992)
5. N. S. Patel, K. L. Hall, K. A. Rauschenbach "40-Gbits cascaded all-optical logic with an ultrafast nonlinear interferometer," *Opt. Lett.* **21** (18), 1466-1468 (1996)
6. S. Nakamura, K. Tajima, and Y. Sugimoto "Experimental investigation on high-speed switching characteristics of a novel symmetric mach-Zehnder all-optical switch," *Appl. Phys. Lett.* **65**, 283-285 (1994)
7. K. I. Kang, I. Glesk, T. G. Chang, P. R. Prucnal, R. K. Boncek "Demonstration of all-optical Mach-Zehnder demultiplexer," *Electron. Lett.* **31** (9), 749-750 (1995)
8. D. Campi, C. Coriasso "Wavelength conversion technologies," *Photonic Netw. Commun.* **2**, 85-95 (2000)
9. I. Glesk, and P. R. Prucnal, "250-Gbps Self-Clocked Optical TDM with a Polarization-Multiplexed clock," *Fiber Integrated Opt.* **14**, 71-82 (1995)
10. K.-L. Deng, R. J. Runser, P. Toliver, C. Coldwell, D. Zhou, I. Glesk, and P. R. Prucnal, "Demonstration of a highly scalable 100-Gbs OTDM computer interconnect with rapid inter-channel switching capability," *Electron. Lett.* **34**, 2418 (1999).
11. P. Toliver, I. Glesk, and P. R. Prucnal, "All-optical clock and data separation technique for asynchronous packet-switched OTDM networks," *Opt. Commun.* **173**, 101-106 (2000)
12. P. Toliver, K.-L. Deng, I. Glesk, and P. R. Prucnal, "Simultaneous Optical Compression and Decompression of 100 Gb/s OTDM Packets Using a Single TOAD and a Bi-directional Optical Delay Line Lattice," *IEEE Photon. Technol. Lett.* **11**, 1183 (1999)
13. K.-L. Deng, R. J. Runser, P. Toliver, I. Glesk, and P. R. Prucnal, "A highly scalable, rapidly-reconfigurable, multicasting-capable, 100-Gbit/s photonic switched interconnect based upon OTDM technology," *J. Lightwave Technol.* **18**, 1892 (2000)

14. K.-L. Deng, R. J. Runser, I. Glesk, and P. R. Prucnal, "Demonstration of Multicasting in a 100-Gb/s OTDM Switched Interconnect," *IEEE Photon. Technol. Lett.* **12**, (5), 558-560 (2000).
  15. K. L. Hall, B. S. Robinson "Bit error rate characterisation of 100-Gbps all-optical demultiplexers," CTuW1 CLEO '99, (1999)
  16. P. Toliver, R. J. Runser, I. Glesk, P. R. Prucnal "Comparision of three nonlinear interferometric optical switch geometries," *Opt. Commun.* **175**, 365-373 (2000)
  17. K. I. Kang, T. G. Chang, I. Glesk, P. R. Prucnal "Comparison of Sagnac and Mach-Zehnder ultrafast all-optical interferometric switches based on a semiconductor resonant optical nonlinearity," *Appl. Opt.* **35** (3), 417-426 (1996)
  18. R. J. Runser "Interferometric SOA-Based Optical Switches for all-optical processing in communication networks and sampling systems," Department of Electrical Engineering, Princeton University, June 2001, Chapter 2 p. 45-47
  19. Y. Ueno, S. Nakamura, K. Tajima "Penalty-free error-free all-optical data pulse regeneration at 84 Gb/s by using a symmetric-Mach-Zehnder-type semiconductor regenerator," *IEEE Photon. Technol. Lett.* **13**, 469-471 (2001)
  20. C. Joergensen, S. L. Danielsen, T. Durhuus, B. Mikkelsen, K. E. Stubkjaer, N. Vojdani, F. Ratovelomanana, A. Enard, G. Glastra, D. Rondi, R. Blondeau "Wavelength conversion by optimized monolithic integrated Mach-Zehnder interferometer," *IEEE Photon. Technol. Lett.* **8**, 521-523 (1996)
- 

## 1. Introduction:

Ultrafast optical demultiplexers are essential components of optical time division multiplexed (OTDM) networks operating at 100 Gb/s and faster. Present approaches to optical demultiplexing include using switches based on soliton gates [1], nonlinear loop mirrors [2], Terahertz Optical Asymmetric Demultiplexer (TOAD) [3], Semiconductor Laser Amplifier in a Loop Mirror (SLALOM) [4], Ultrafast nonlinear interferometer (UNI) [5], Mach-Zehnder interferometer with semiconductor optical amplifier [6,7], and Michelson interferometer with semiconductor optical amplifiers [8]. Due to the simplicity of design and low switching energy, the TOAD has been used in numerous OTDM systems and network demonstrations [9-14]. The TOAD consists of an optical loop mirror with an SOA placed off centered in the loop. The offset from the center determines the switching window width of the TOAD, as the asymmetry leads to a difference in arrival time between the two counter propagating data pulses at the SOA. A precisely timed clock pulse is injected into the loop such that its arrival at the SOA provides a relative  $\pi$  phase shift to the data pulse entering the SOA after the clock pulse has hit the SOA. This results in constructive interference at the output of the TOAD and the output pulse emerges.

One limitation of the TOAD approach is the finite propagation time of the pulse across the SOA [15]. If the offset of the SOA from the center is decreased such that the SOA starts to straddle the center of the loop, the effective SOA length that the two counter propagating pulses see is reduced. The decrease in effective SOA length leads to a reduction in the contrast ratio of the TOAD switching and thus, an excess power penalty. The effective length of the SOA required for producing the relative  $\pi$  phase shift places a practical limitation on the switching window size of the TOAD to be greater than the propagation time of the pulse through the SOA. The smallest switching window size obtained with a TOAD is 3.5ps [16].

In this paper, we demonstrate a new method of achieving narrow switching windows by cascading two TOADs, each with the SOA on opposite sides of the fiber loop. This method overcomes the limitation on the switching window placed by the length of the SOA.

## 2. Principle of operation

A characteristic of the TOAD switching window is that the rising and falling edges have different slopes [16]. The location of each edge is determined by the side of the fiber loop that the SOA is placed with respect to the control port. If the SOA is placed on the same side of the loop as the control port then the rising edge of the switching window is very steep, limited

only by the clock pulse width. The falling edge slope of the switching window is a result of the clock and the counter propagating data pulses meeting inside the SOA and is thus related to the propagation time of the pulse through the SOA. If the SOA is placed on the opposite side of the loop, as the control port, the two edges are interchanged and the falling edge of the switching window is much steeper than the rising edge. Figure 1 shows the switching windows with the SOAs on different sides of the loop.

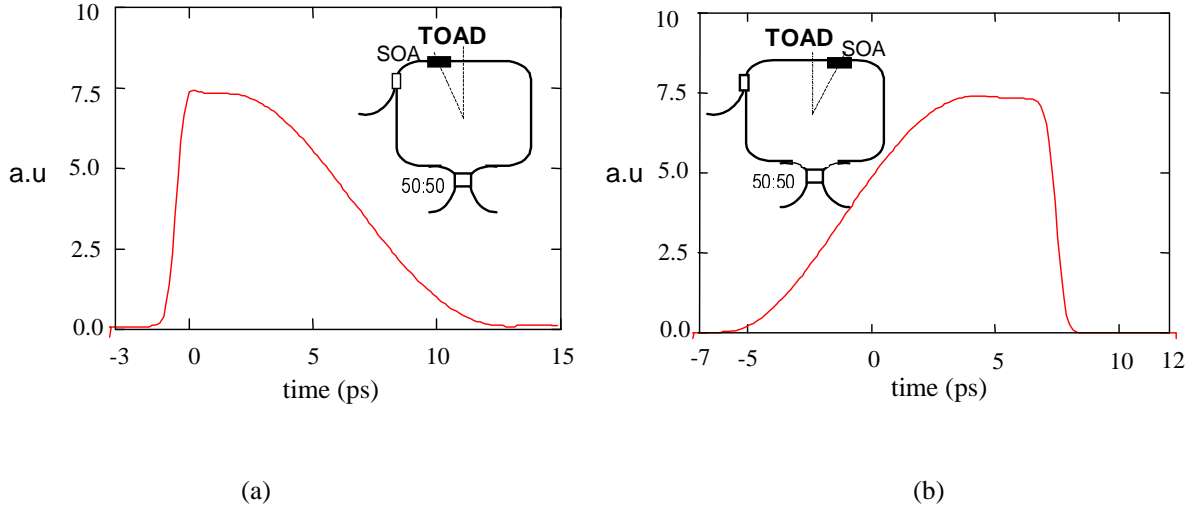


Fig 1: The switching windows with the SOA on (a) same side as the control port (b) different side as the control port

The transfer function of one TOAD can be written by the basic interferometric equation [17]

$$SW(t) = \left\{ G_1(t) + G_2(t) - 2\sqrt{G_1(t)G_2(t)} \cos(\phi_1(t) - \phi_2(t)) \right\} \quad (1)$$

where  $G_1(t)$ ,  $\phi_1(t)$  and  $G_2(t)$ ,  $\phi_2(t)$  are the time dependent gain and phase changes experienced by the two counterpropagating pulses as they traverse the SOA. By cascading the two TOADs with a time shift of  $\delta$  between the two windows, the overall transfer function referred to as *Cascade* ( $t, \delta$ ) becomes the product of the two constituent ones,  $SW_A(t)$  and  $SW_B(t)$  with the time shift taken into account.

$$Cascade(t, \delta) = SW_A(t) \times SW_B(t - \delta) \quad (2)$$

In such a configuration, one TOAD has the SOA on the same side of the loop as the control port and the other has the SOA on the opposite side of the loop as the control port. Their switching windows are then placed such that the sharp edge of each overlaps the sharp edge of the other, as shown in figure 2, and this results in a switching window size limited only by the optical pulse width of the clock and data.

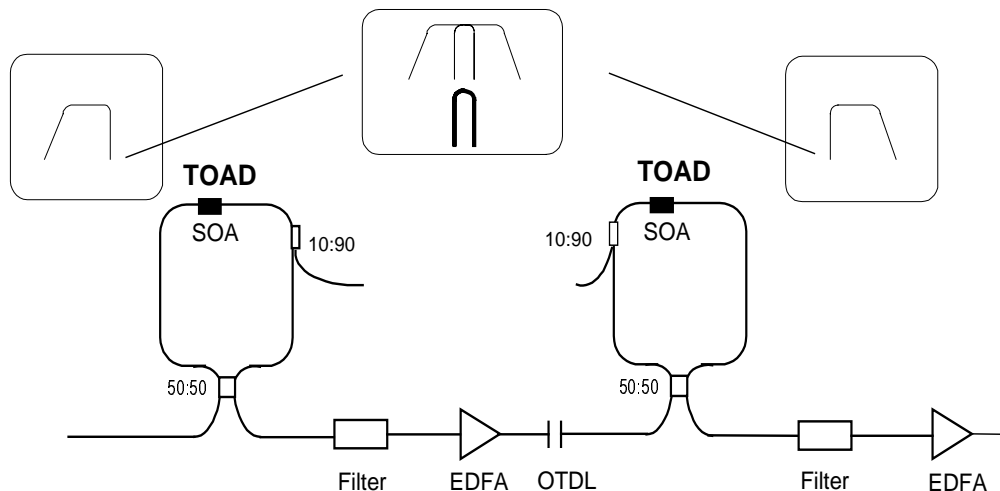


Fig 2 The principle of the optical switch based on overlap of two switching windows

Thus, in comparison with the individual switching windows of figure 1, the cascaded TOAD has its switching window's falling and rising edges determined by the sharp edges of the two individual windows. The slow falling and rising edges of the two individual TOAD windows, limited by the propagation time of the pulses in the SOA, do not affect the switching window of the cascaded TOAD.

To study this effect, simulations were done using a model for the gain and the phase changes, based on previous work [17]. The simulation uses Gaussian pulses with 1ps width for input clock and 1.5 ps width for the input data. The SOA is taken as 500  $\mu\text{m}$  long with a 200ps recovery time. Figure 3 shows the simulated switching window of one TOAD for different SOA offsets. As the SOA is moved from one side of the loop to the other, the rising and falling edges are interchanged. In figure 4, the switching window of the cascaded TOADs is simulated with different delay offsets  $\delta$ , between the two windows each of which are 8ps wide. The switching window amplitude remains fairly constant until the width is decreased to 1.4ps.

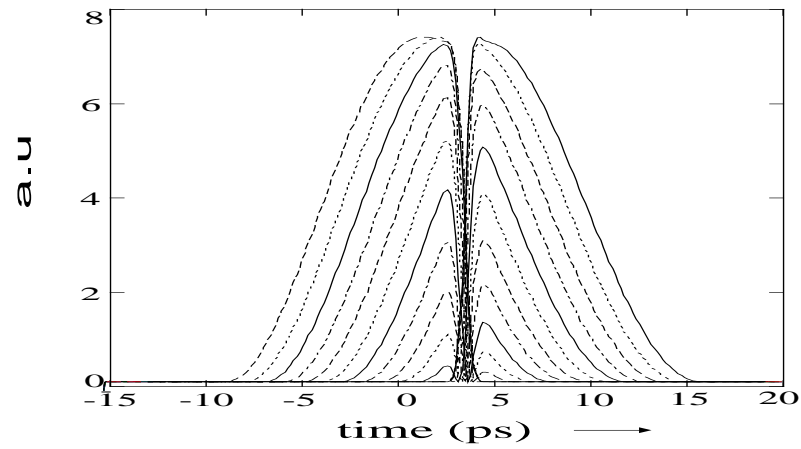


Fig 3: The simulated switching window of a TOAD for different SOA offsets

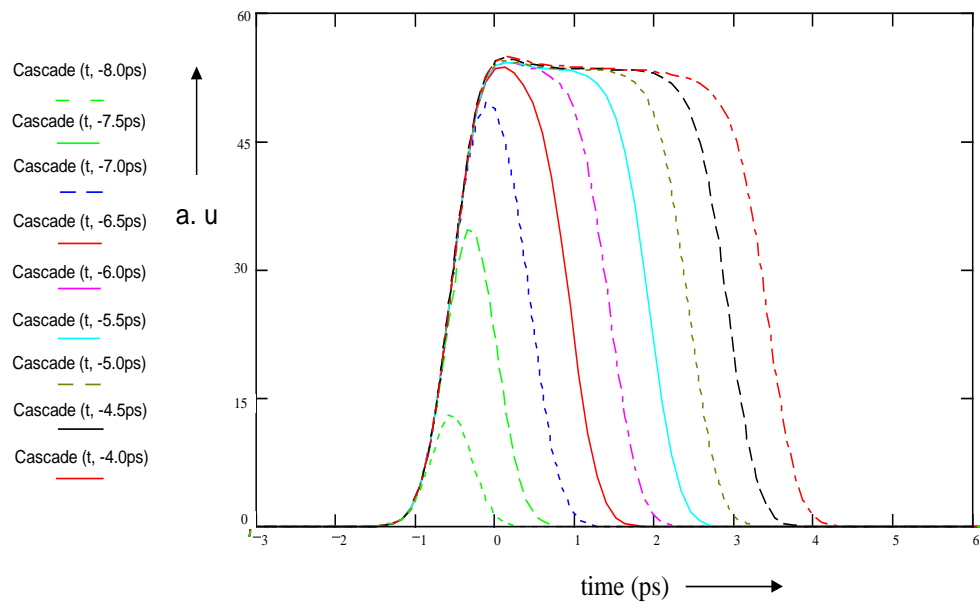


Fig 4: The simulated switching windows with different delay offsets  $\delta$  between the two TOADs

### 3. Experiments and results:

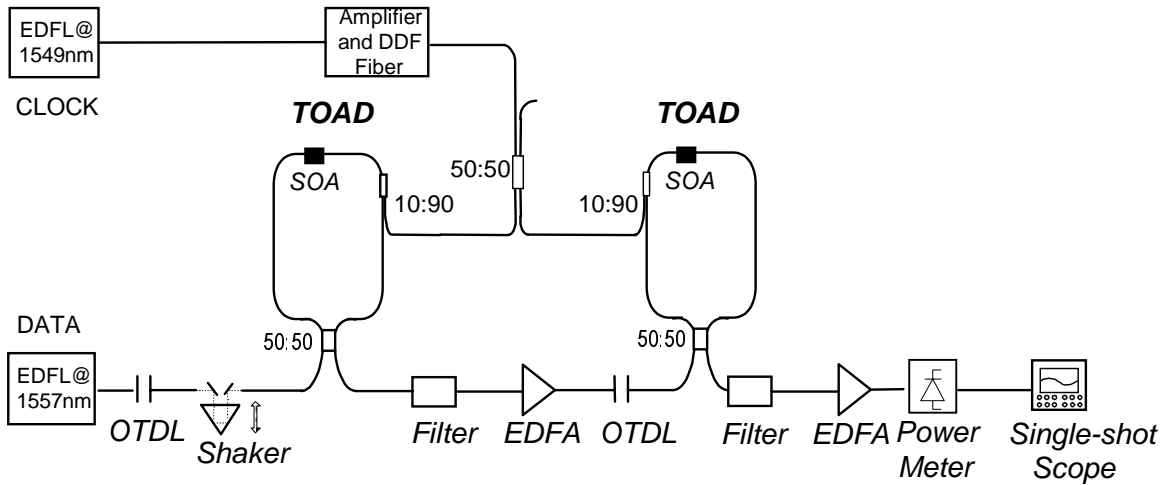


Fig 5: The experimental setup to measure the switching window of the cascaded TOAD

EDFL: Erbium Doped Fiber Laser, OTDL: Optical Tunable Delay Line, DDF: Dispersion Decreasing Fiber

EDFA: Erbium Doped Fiber Amplifier, SOA: Semiconductor Optical Amplifier

The experimental demonstration, shown in figure 5, uses two TOADs, each with an 8ps switching window but with the SOA on opposite sides of the fiber loop with respect to the control port. Two mode-locked Erbium Doped Fiber Lasers are used as clock and data pulse sources. The SOA used in the experiment is a 500 $\mu$ m long Alcatel 1901 SOA biased near 100mA. The clock pulses have a wavelength of 1549nm and pass through an amplifier and dispersion-decreasing fiber, which compresses the pulse width to 1ps. The data pulse width is 1.5ps and the wavelength is 1557nm. Prior to entering the first TOAD, the data pulse first passes through a free space delay stage and a mechanical shaker. The free space delay stage positions the temporal position of the data in the range of the switching window of the cascaded TOAD. The mechanical vibrator is used to quickly scan the data pulses in time over a 30ps range to map out the switching window. By continuously monitoring the switching window on the oscilloscope, this technique provides a means of rapidly characterizing the switching window [18]. After the mechanical vibrator, the data pulses enter the first TOAD. The data pulse then exits from the output of the first TOAD and passes through a filter, an EDFA and a tunable free space delay line, before entering the second TOAD. The filter rejects the clock pulse of the first TOAD from entering the second TOAD. The tunable delay line between the two TOADs controls the temporal position of the second TOAD's switching window relative to the switching window of the first TOAD, and thus provides the control over the aggregate switching window size of the two cascaded TOADs. The output of the second TOAD first passes through an optical filter to reject the clock pulse before entering a power meter. The power meter is connected to a single shot oscilloscope that is synchronized to the mechanical shaker. The switching window size can be inferred because the oscilloscope displays the convolution of the data pulse with the switching window of the cascaded TOADs.

Figure 6 shows experimental scan of the switching windows of the cascaded TOADs for different delay times  $\delta$  between the two TOADs as it appears on the single shot sampling scope. The shape of the experimental switching windows differs from the simulated ones.

This could be due to experimental non-idealities like variation in control pulse energy and slight variations in coupling losses of the mechanical vibrator used to scan the data signal through the switching window. Also, the simple simulation model may not adequately take into account other physical device level effects that may influence the switching behavior.

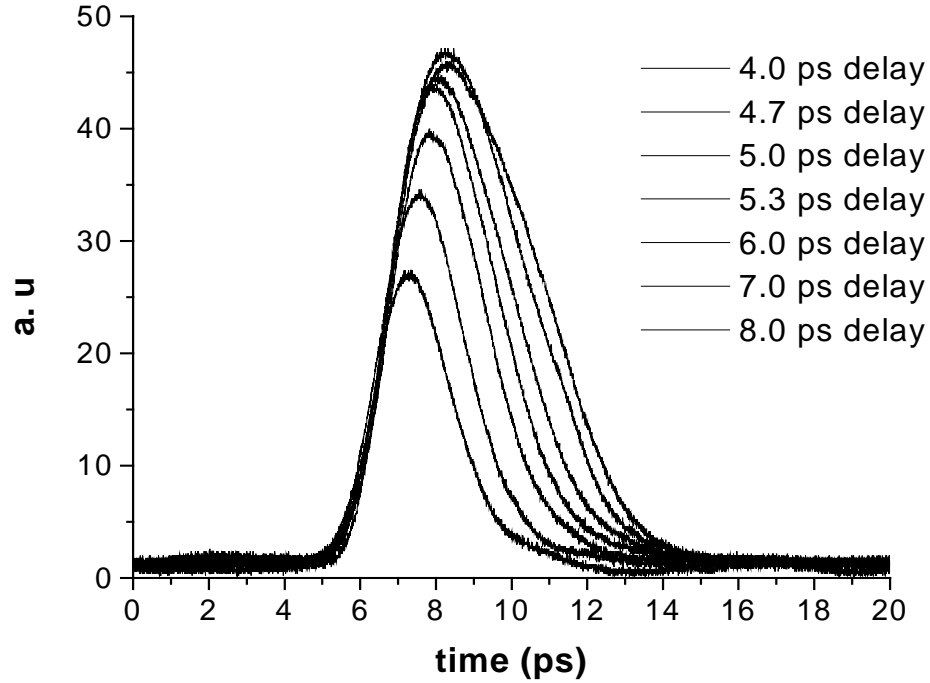


Fig 6: Experimental data, showing the switching windows convolved with the data pulses, for different delay offsets  $\delta$ , between the constituent switching windows

The experimental windows shown in figure 6 represent the convolution of the 1.5ps data pulses with the actual cascaded TOAD switching window. The actual switching window size can be inferred by de-convoluting the measured switching window with a 1.5ps pulse. We compare the actual switching window widths obtained theoretically and experimentally as a function of  $\delta$ , the delay offset between the two TOADs, in figure 7. We also show the switching windows obtained after de-convolution with the data pulse. The difference between the experimental and de-convoluted experimental switching windows increase as the switching window becomes smaller. The cascaded TOAD switching window is also limited by the input clock pulse width; this is evident in the decrease in switching window amplitude when the window width narrows to less than 2.9 ps.

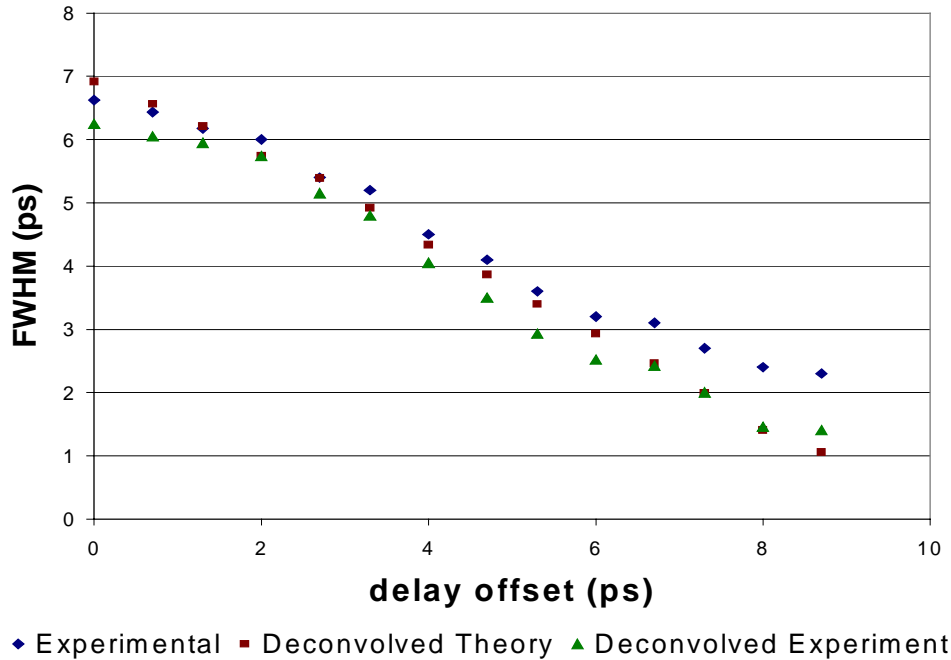


Fig 7: The full width half maxima of the experimental, theoretical and de-convoluted experimental switching windows with different delay offsets,  $\delta$

#### 4. Discussion:

Presently, one of the best performing interferometric fiber-based optical switch geometry is the Symmetric Mach-Zehnder (SMZ) geometry [16], and we compare the performance of the cascaded TOAD to the CMZ. The SMZ requires stringent polarization control as thermal fluctuations can offset the interference conditions between the two arms [18]. The cascaded TOAD is immune to thermal effects in the laboratory because the two counter propagating pulses in each TOAD travel through the same span of fiber. However, one advantage that the SMZ structure enjoys over the cascaded TOAD is that the SMZ can be easily integrated onto a single photonic chip. Various wavelength converters and 3R regenerators have already made use of this and similar structures [19,20]. The integration of the cascaded TOAD is presently a difficult problem yet to be solved.

The drift in delay offset between the two individual TOADs due to thermal fluctuations can lead to slight variations in overall switching window size. However, this is of the order of few femtoseconds and is negligible compared to the overall switching window size, which is of the order of picoseconds. For constant switching window size, the delay line between the individual TOADs can be replaced by fusion spliced fiber lengths. This will reduce the losses incurred by the signal on passing from one TOAD to another, resulting in higher switching contrast ratio.

The overall switching window depends only on the extent of partial overlap between the two switching windows and not on the relative sizes of the individual windows. However, it is necessary to ensure that the smaller window does not fall completely within the larger window, in which case the overall switching window will be the same as the smaller window and will be limited by the propagation time of pulses in the SOA. Hence, the cascaded TOAD



gives the same characteristics for different relative locations of SOAs in the loops, when the overlap is partial.

In summary, we demonstrated a novel switch based on two TOADs by utilizing the switching window characteristics of each to achieve a narrower temporal switching window output.

# All-Optical Clock Division With Mode-Locked Figure-Eight Laser Based on the Slow Carrier Recovery Rate in Semiconductor Optical Amplifier

Lei Xu, Minyu Yao, Bing C. Wang, Ivan Glesk, *Senior Member, IEEE*, and Paul R. Prucnal, *Fellow, IEEE*

**Abstract**—With a mode-locked figure-eight laser, we demonstrate the clock division operation using a semiconductor optical amplifier (SOA) with a slow carrier recovery rate. We show stable clock division at 2.5 GHz and two output states at half the repetition rate of the 10-GHz input pulses. The forming pulses in the laser cavity will interact with each other at the onset of mode locking when the SOA in the laser has a relatively slow carrier recovery rate. A numerical model of the laser is built to simulate the generation of clock division.

**Index Terms**—Mode-locked lasers, optical communication, semiconductor optical amplifiers, time-division multiplexing.

## I. INTRODUCTION

IN OTDM systems, separate channels at the same frame rate are multiplexed together in the time domain into a high-speed aggregate data stream. To access the individual time slots, high-speed all-optical clock division can be used to generate optical clock signal at the frame rate for time-domain demultiplexing. Various all-optical clock division schemes have been demonstrated [1], [2]. Manning *et al.* demonstrated all-optical clock division based on the configuration of terahertz optical asymmetric demultiplexer (TOAD) with a feedback loop [1], and pointed out that stable mode of clock division occurs when the input pulse repetition rate approaches the gain recovery rate of the semiconductor optical amplifier (SOA) [3]. Lee demonstrated a polarization-independent all-optical SOA/grating-filter switch for clock division using self-phase modulation in the SOA [2]. In Manning's and Lee's approaches, an SOA with very short lifetime is required for OTDM data streams with high line rate. In this letter, we demonstrate optical clock division with a mode-locked figure-eight laser that uses a TOAD as an optical modulator. A laser with similar structure was previously proposed for clock recovery [4]. We show that neighboring pulses in the laser cavity will interact with each other at the onset of the mode locking process, and that all-optical clock division occurs by using an SOA that has a recovery rate much slower than the line rate of the input optical pulses. We also build a numerical model of the laser to simulate the generation of clock division,

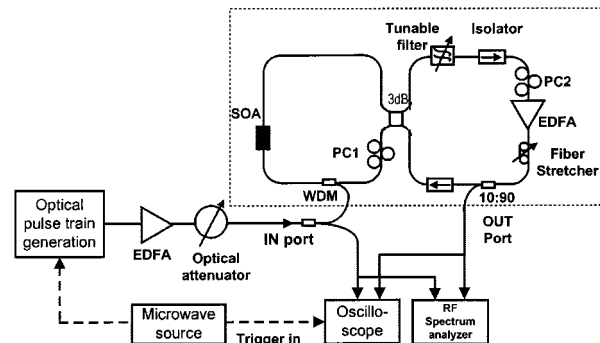


Fig. 1. Experimental setup.

and analyze the working modes of the laser for different conditions.

## II. EXPERIMENTS AND RESULTS

The experimental arrangement is shown in Fig. 1. The laser is composed of two loops: the left loop with an SOA has a TOAD structure and functions as an optical modulator; the right loop with the optical isolators functions as a ring cavity. A fiber stretcher in the right loop adjusts the laser cavity length to satisfy the requirement for harmonic mode locking. The tunable band-pass filter (BPF) has a bandwidth of 1.0 nm. An erbium-doped fiber amplifier (EDFA) provides the gain. The total cavity length is about 60 m. An optical attenuator before the "IN" port is used to control the input optical pulse power. The experimental results at 25 and 10 GHz are discussed separately.

1) At 2.5 GHz, the switching pulses are derived from a gain-switched distributed-feedback (DFB) laser diode at 1534 nm and then compressed to 12.0 ps by dispersion compensation fiber. The tunable filter in the laser cavity is centered on 1557 nm.

The trace and RF spectrum of the input signal are shown in Fig. 2. When the injection current of SOA is larger than 130 mA, the laser can be mode locked by the input pulse train and the output pulses have the same repetition rate as the input pulses. In order to get the stable operation of clock division, we set the injection current of SOA to 110 mA with the associated gain recovery time to be about 700 ps, and optimize the optical power of the input pulses and the loop gain in the laser cavity. When the mean optical power of the injected pulses is 8 mW, the stable output at half the repetition rate of the input signal arises and the mean optical power in the laser cavity is 5 mW. Fig. 3 shows the

Manuscript received August 16, 2001; revised October 4, 2001.

L. Xu, B. C. Wang, I. Glesk, and P. R. Prucnal are with the Department of Electrical Engineering, Princeton University, NJ 08544 USA (e-mail: Leixu@ee.princeton.edu).

M. Yao is with the Department of Electronic Engineering, Tsinghua University, 100084 Beijing, China.

Publisher Item Identifier S 1041-1135(02)00874-1.

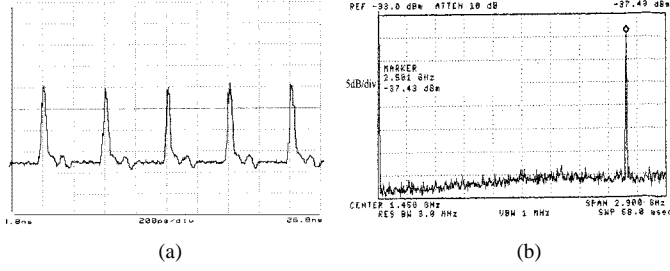


Fig. 2. (a) Oscilloscope traces (200 ps/div), and (b) radio frequency spectrum of the input signal at 2.5 GHz.

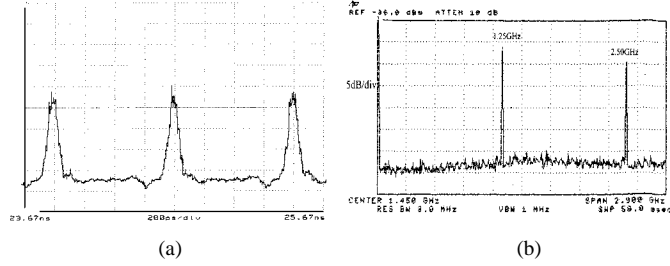


Fig. 3. (a) Oscilloscope traces (200 ps/div), and (b) radio frequency spectrum of the output signal at half the repetition rate of the input signal.

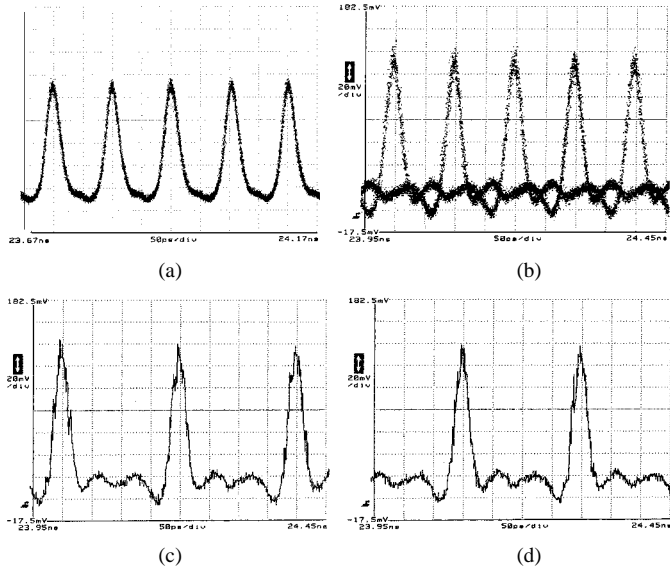


Fig. 4. (a) Input 10-GHz pulse trace, (b) Output signal from the laser, (c) Quasi-stable state I, (d) Quasi-stable state II (horizontal scale is 50 ps/div).

trace and RF spectrum. In Fig. 3(b), the sharp spectral component at half the frequency of the input signal clearly shows the clock division operation.

2) To perform clock division at 10 GHz, we use a 10-GHz erbium-doped fiber laser as the pulse source. The input pulsewidth is 11 ps and the average optical power is 12 mW. The input pulse trace is shown in Fig. 4(a). The optical power inside the laser cavity is 6 mW. When the current is set to 120 mA, we get the “eye diagram”-like output signal shown in Fig. 4(b). After decreasing the persistent time of the oscilloscope display, we find that the trace in Fig. 4(b) is a combination of two quasi-stable states shown in Fig. 4(c) and (d). The two states, which are at half repetition rate of the input, have a half period timing difference. In our experiments, each of the states can be stable for

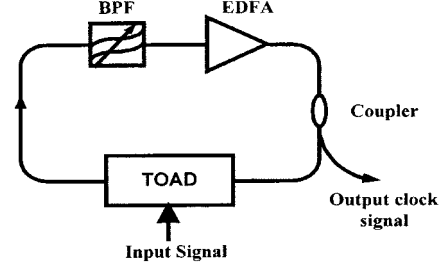


Fig. 5. Numerical model of the figure-eight laser.

1–2 s. For 10 GHz, the recovery time of the SOA used may be too long, which may result in unstable operation. The unstable output of the laser could also be the result of cavity length fluctuation from imperfect laboratory conditions, such as thermal fluctuations or mechanical relaxation of the fiber stretcher.

In our experiments, the working conditions of the laser, including the SOA bias current, optical power of the input signal and cavity light must be optimized to generate the clock division operation.

### III. WORKING PRINCIPLES

The generation of clock division in the figure-eight laser is directly related to the working conditions of the TOAD. During the startup of the mode locking process, an input pulse passing through the TOAD saturates the SOA and the light in the laser cavity experiences a switching window. Due to the slow carrier recovery rate, every subsequent pulse will experience a smaller transmission coefficient through the TOAD than the previous one. The interaction between adjacent pulses from the long SOA recovery time will be further enhanced by the mode locking mechanism. Through the combined effects of pulse interaction and mode locking, the figure-eight laser generates a clock divided signal at half the frequency of the entering pulse.

We build a numerical model to illustrate the generation of clock division. Fig. 5 shows the diagram of the laser model. In the simulation, the SOA carrier recovery rate is 400 ps and the input signal is a 10-GHz pulse train, where the pulsewidths are 10 ps and peak power is 100 mW. The average power of the light in the laser cavity is 4 mW. The evolution of optical pulses in the cavity during the start-up process is shown in Fig. 6. The left column of Fig. 6 is the pulse traces, showing the width and the amplitude of the pulses. The right column shows the spectra of the pulses.

In the beginning stages, shown in Fig. 6(a) and (b), the repetition rate of the pulses in the laser cavity is 10 GHz. The spectrum shows that the spacing between neighboring lines is 10 GHz. The pulse amplitude variations in the time trace are caused by the slow SOA carrier recovery rate. Since the carrier recovery time of the SOA is longer than the time between adjacent pulses, optical modulation by the input pulses will not be the same for all the pulses in the laser cavity during the startup of mode locking. In stages  $N = 50$  to  $N = 70$ , shown in Fig. 6(c) and (d), the amplitude differences between the pulses in the cavity are further enhanced, and another sharp spectral lines appear between the neighboring 10 GHz lines in their spectra. Due to interactions with the input pulses, stronger pulses in the cavity will

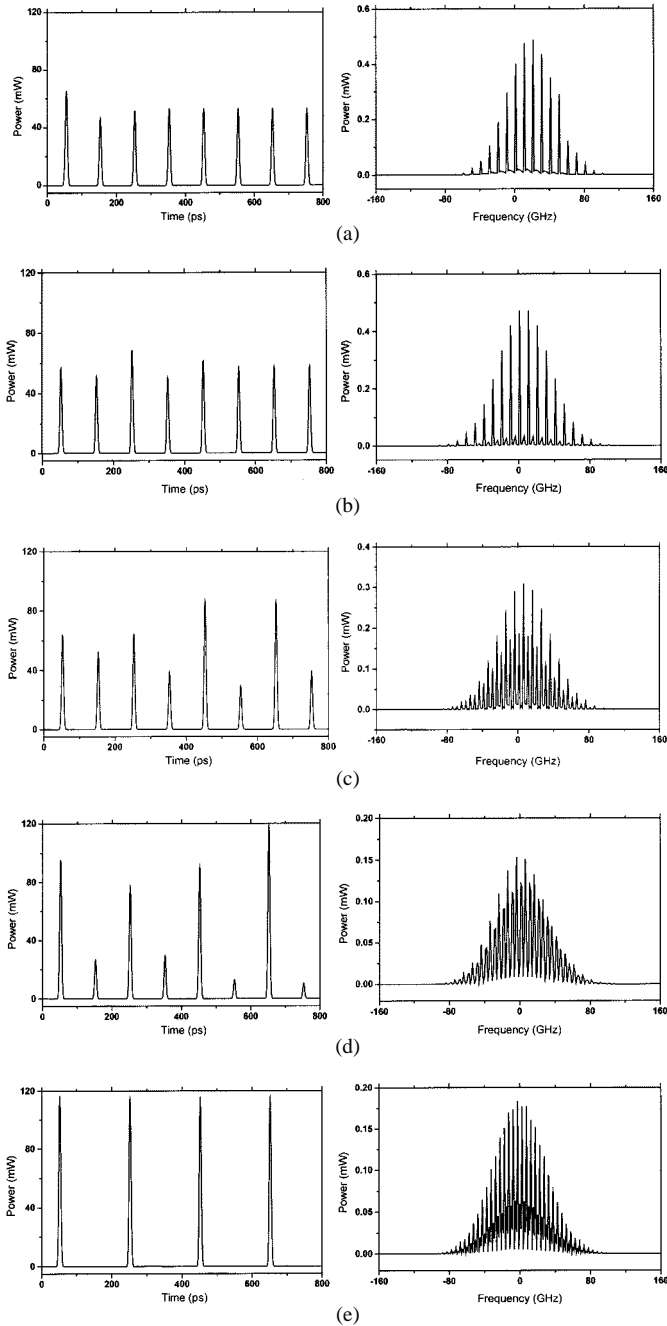


Fig. 6. The startup process of the clock division operation (a)  $N = 3$ , (b)  $N = 20$ , (c)  $N = 50$ , (d)  $N = 70$ , and (e)  $N = 140$  ( $N$  is the number of calculation circles).

experience a higher transfer coefficient when passing through the TOAD than weaker pulses. This process further enhances strong pulses and suppresses weak pulses. After the start-up process, the laser can work in a stable clock division state, as is shown in Fig. 6(e). The spectrum shows that the spacing between neighboring lines is 5 GHz.

Fig. 7 shows the influence of the SOA recovery time and the input pulse power on the working range of the clock division operation at 10 GHz. The area in which clock division can occur is labeled area "B" in Fig. 7 and this area approximately stretches from a carrier lifetime range of 350–500 ps and peak power levels of 0.05–0.15 W. In region B, the neighboring pulses will

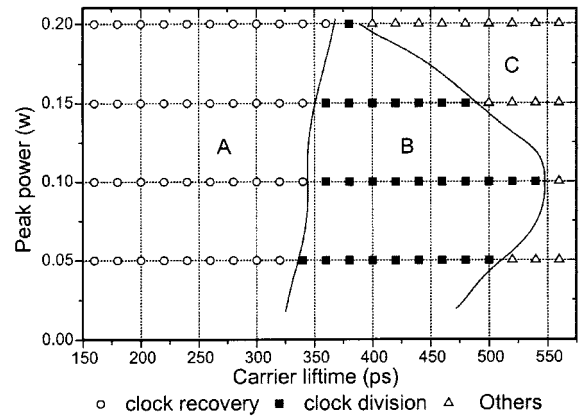


Fig. 7. Working status of the laser under different SOA carrier recovery time and peak power of the input pulse.

have strong interactions with each other during the startup of mode locking, which results in clock division.

In region A, the SOA has a relatively short recovery time. The interactions through the TOAD between optical pulses in the laser cavity are weak. The output pulses repetition rate will be equal to the input, which is used for clock recovery. In region C, the carrier lifetime is further increased where not just two pulses, but multiple neighboring pulses in the laser cavity will interact with each other due to the very slow carrier recovery rate. As a result, the figure-eight laser can produce a signal at one fourth or one eighth of the repetition rate of the input signal. However, in practical applications, stability issues limit the number of output pulse rate divisions in which the laser can be operated.

Since the clock-divided signal also satisfies the harmonic mode locking condition, an additional condition is required for clock-division operation: the repetition rate of the input signal should be an even multiple of the round-trip frequency of the laser cavity.

#### IV. CONCLUSION

We demonstrate the optical clock-division operation using a mode-locked figure-eight laser that employs an SOA with a slow carrier recovery rate. We show stable clock division at 2.5 GHz, and two clock division states at half the repetition rate of the 10-GHz input pulses. A numerical model was built to simulate the generation of clock division in the laser. At the onset of mode locking, the forming pulses in the laser cavity will have strong interactions with each other as a result of the slow SOA carrier recovery time, which leads to the generation of clock division.

#### REFERENCES

- [1] R. J. Manning, A. J. Poustie, and K. J. Blow, "All-optical clock division using a semiconductor optical amplifier loop mirror with feedback," *Electron. Lett.*, vol. 32, pp. 1504–1506, 1996.
- [2] H. J. Lee and H. G. Kim, "Polarization-independent all-optical clock division using a semiconductor-optical-amplifier/grating-filter switch," in *OFC'99*, 1999, pp. 95–97.
- [3] R. J. Manning, A. E. Kelly, K. J. Blow, A. J. Poustie, and D. Nasset, "Semiconductor optical amplifier based nonlinear optical loop mirror with feedback: Two modes of operation at high switching rates," *Opt. Commun.*, vol. 157, pp. 45–51, 1998.
- [4] L. E. Adams, E. S. Kintzer, and J. G. Fujimoto, "Performance and scalability of an all-optical clock recovery figure eight laser," *IEEE Photon. Technol. Lett.*, vol. 8, pp. 55–57, Jan. 1996.



ELSEVIER

15 November 2001

OPTICS  
COMMUNICATIONS

Optics Communications 199 (2001) 83–88

www.elsevier.com/locate/optcom

# Experimental study on the regeneration capability of the terahertz optical asymmetric demultiplexer

Bing C. Wang<sup>\*</sup>, Lei Xu, Varghese Baby, Deyu Zhou, Robert J. Runser,  
Ivan Glesk, Paul R. Prucnal

*Princeton University, J303 E-Quad, Princeton, NJ 08544, USA*

Received 1 August 2001; accepted 6 September 2001

## Abstract

All-optical regeneration capability of the terahertz optical asymmetric demultiplexer (TOAD) is experimentally explored. A continuous wave light is injected into the TOAD structure to reduce the pattern dependent effect caused by the data pulse injected into the control port. The bit error rate (BER) of the regenerated output at 10 Gb/s is measured with different amounts of input data timing jitter and switching window size. Experimental results show that optical regeneration using the TOAD has a strong tolerance to the timing jitter of the incident data signal. Open eye diagram and BERs below  $10^{-9}$  have been successfully obtained for input timing jitter value of 17 ps. © 2001 Elsevier Science B.V. All rights reserved.

**Keywords:** Regeneration; OTDM; All-optical

## 1. Introduction

In optical time division multiplexing (OTDM) networks and transmission systems, individual channels are assigned unique time slots and multiplexed to generate high-speed data streams. Precise control of the timing of data pulses is critical for proper multiplexing and demultiplexing with minimal inter-channel crosstalk. In OTDM systems, the optical pulse width occupies close to a third of the inter-channel spacing between adjacent time slots in the aggregate data stream, so strict control of the data pulse shape and pulse

width is required throughout the network. These requirements to maintain the timing and pulse shape within the network while retaining the data in the optical form motivate the development of all optical retiming, reshaping and reamplifying (3R) regenerators.

In optical 3R regeneration, optical clock pulses are modulated by the incident data signal through a nonlinear optical gate. The core of an optical regenerator is the nonlinear gate that provides noise reduction, timing jitter suppression, and reshaping capability. Semiconductor-based optical gates attract special interest [1–4] since they are compact and require low switching energy. The terahertz optical asymmetric demultiplexer (TOAD) is an interferometric fiber loop mirror based optical gate with a semiconductor optical amplifier (SOA)

<sup>\*</sup> Corresponding author. Tel./fax: +1-609-258-2041.

E-mail address: bingwang@princeton.edu (B.C. Wang).

placed asymmetrically in the loop [2]. The TOAD has already been used for OTDM demultiplexing [5], optical signal processing [6], optical packet switching [7], and optical memory [8]. For all optical regeneration, jittered data pulses are injected into the control port of the TOAD. One problem associated with regeneration using the TOAD is that the recovery time of the SOA may place limitations on the optical data bit rate due to pattern dependent effects. In this paper we use a continuous wave (CW) pump light to enhance the recovery rate of the SOA to suppress pattern dependent effects [9–13]. The optical regeneration capability of the TOAD at 10 Gb/s is measured by changing the timing jitter of input data pulses and the size of the switching window. Experimental results show that optical regeneration using the TOAD has a strong tolerance to the timing jitter of the incident data signal.

## 2. Principle of operation

The optical regenerator based on the TOAD is shown in Fig. 1. The SOA is placed asymmetrically in the optical loop. The data pulse enters the TOAD through the 90/10 coupler and opens a switching window in the optical regenerator. The optical tunable delay line in one arm of the loop controls the offset position of SOA, which determines the temporal width of the switching window. Strong data pulses control the switching of the TOAD through the nonlinear effect in the SOA. However, at high bit rates, the slow SOA recovery will cause 1 bit preceded by 0 bit to experience more phase change than 1 bit preceded by

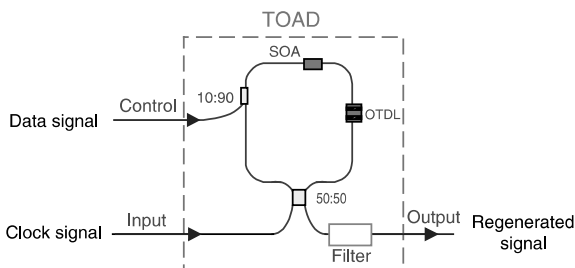


Fig. 1. Optical regeneration with TOAD.

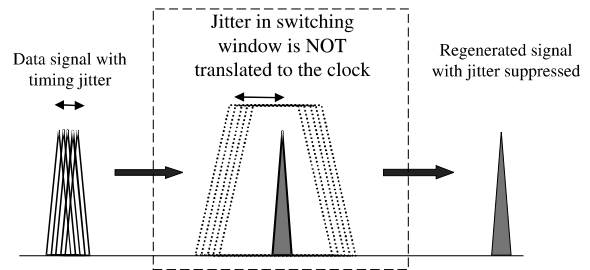


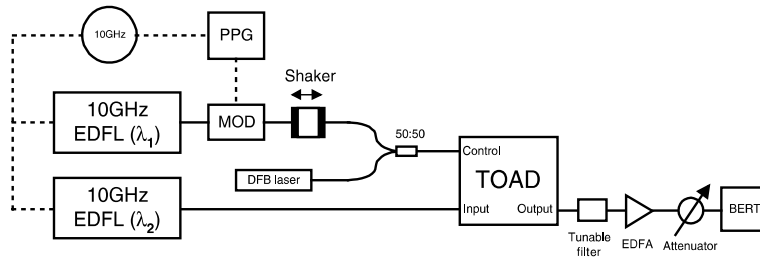
Fig. 2. Timing jitter suppression through regeneration using TOAD.

another 1 bit. The effect of this pattern dependency will reduce the extinction ratio of the retimed data and close the eye diagram. Enhancing the SOA recovery rate can reduce this pattern dependent effect.

Because the timing of the clock pulses is not affected by the location of the switching window in which they reside, the pulses that are switched out will have the information that the input data pulses have, but none of the timing jitter. This process is shown in Fig. 2. However, when the timing jitter of the incident data pulses exceeds the tolerance of the optical regenerator, the timing jitter of incident data pulses will be converted to the relative intensity noise. We measure the bit error rate (BER) for different switching window size as well as different timing jitter value to gain a better understanding of the TOAD's performance characteristics.

## 3. Experiments and results

The optical regeneration experimental setup is shown in Fig. 3. Two 10 GHz mode-locked erbium doped fiber lasers generate 2 ps pulses for the data and clock, the data at  $\lambda_1 = 1548$  nm and the clock at  $\lambda_2 = 1554$  nm. The data is modulated with  $2^{31}-1$  pseudorandom bit sequences and the timing jitter is added using a free space shaker with an attached retro cube. A distributed feedback (DFB) laser at 1539 nm provides the pumping light to enhance the SOA carrier recovery rate. The output of the DFB laser is combined with the data pulses through a 50:50 coupler and then injected into the optical regenerator through the control port.



EDFL: Erbium doped fiber laser; PPG: pseudorandom pattern generator;  
MOD: modulator; BERT: bit error rate tester; DFB: distributed feedback,

Fig. 3. Schematic setup of optical regeneration using TOAD.

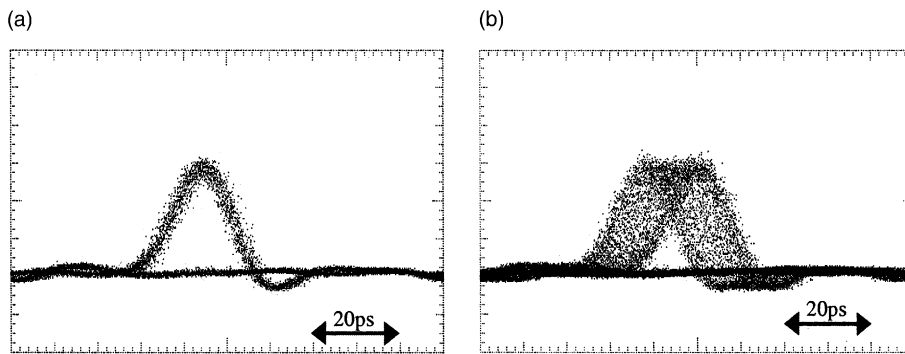


Fig. 4. The eye diagram of (a) generated data signal after modulator and (b) data signal with timing jitter added.

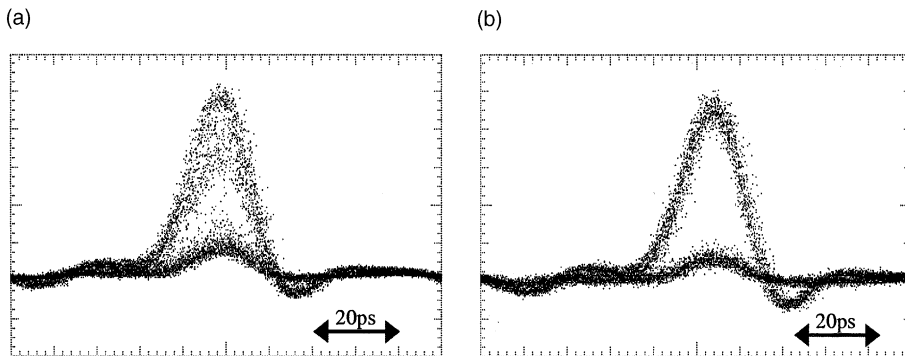


Fig. 5. Regenerated signal (a) without pumping light injection and (b) with pumping light enhancing SOA carrier recovery.

Fig. 4(a) shows the input data without the added timing jitter, and Fig. 4(b) shows the input data with a timing jitter of 17 ps. The timing jitter is measured at peak-to-peak values. Fig. 5(a) shows the eye diagram of the output from the TOAD

without the pump light injection. Compared with Fig. 4(b), Fig. 5(a) shows the timing jitter of the input data dramatically suppressed after passing through the TOAD. However, the output shows poor extinction ratio and large intensity noise due

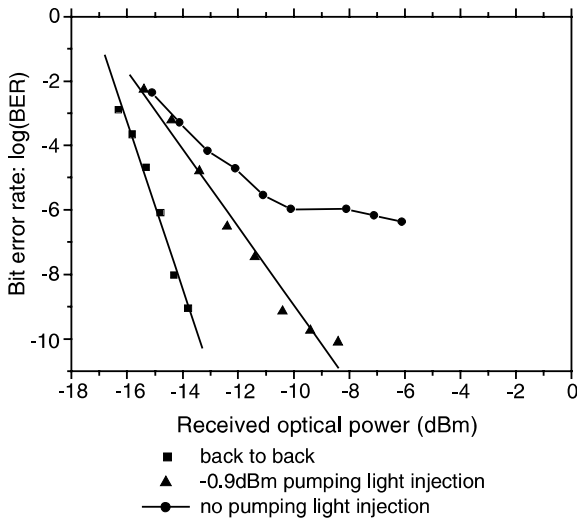


Fig. 6. Measured BER with/without pumping light.

to the pattern dependent effect. To eliminate the 17 ps of timing jitter in the input data, the switching window of the TOAD is set to 46 ps. The SOA current is set to 180 mA, and the carrier recovery rate is longer than 100 ps. For Fig. 5(a), the data pulses entering the control port of TOAD have pulse energy of 300 fJ. With pump light injection of  $-0.9$  dBm, the regenerated signal is shown in Fig. 5(b). Comparing Fig. 5(a) and (b), the pattern dependent effect is dramatically suppressed by pump light injection, and both figures show elimination of input data timing jitter. In the measured BER curves in Fig. 6, there is a BER floor at around  $10^{-6}$  for the regeneration without pumping light, where as  $10^{-10}$  BER is achieved with the injection of pumping light.

To understand the amount of incident data timing jitter the TOAD can remove, we measure the output BER at different input timing jitter amplitudes. Fig. 7 compares BER curves from three different timing jitter values: 10, 17 and 21 ps, all with CW injection power of  $-0.9$  dBm. The size of the switching window remained at 46 ps. The power penalty decreases approximately 0.5 dB if the timing jitter is reduced from 17 to 10 ps. When we increase the timing jitter to 21 ps, the BER curve degrades significantly such that a BER floor appears near  $10^{-8}$ . From Fig. 2, we can see that the timing jitter of the incident data pulses translates

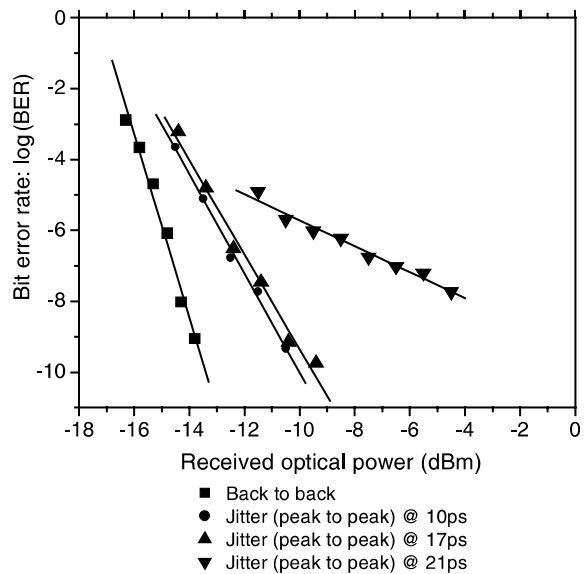


Fig. 7. Measured BER under different timing jitter of the incident data pulses.

to a timing jitter in the TOAD switching window. Fig. 8 shows that when the timing jitter exceeds the switching window size of the TOAD, the excess timing jitter will be converted to intensity noise at the output.

When the timing jitter of the incident data pulses is within the tolerance of the regeneration capability of TOAD, there is a slight difference of power penalty for different timing jitter input.

As shown in Fig. 2, the shape and size of switching window of the optical regenerator is critical to the performance of the regeneration. When the control pulse is narrow, TOAD has a square-like switching window with a flat top [14]. Fig. 9 shows the BER curves under different switching window sizes: 46, 40 and 32 ps. The pumping beam is kept at the power level of  $-0.9$  dBm, and the input data pulses have a timing jitter of 10 ps. When the window size is decreased, the power penalty increases. For the window size of 32 ps, error floor near  $10^{-4}$  appears. For optical regeneration, a wide and flat window will increase the regeneration capability. However, the maximal size of the window is limited by the bit rate of the incident data, otherwise an overly large window will cause crosstalk between neighboring bits.



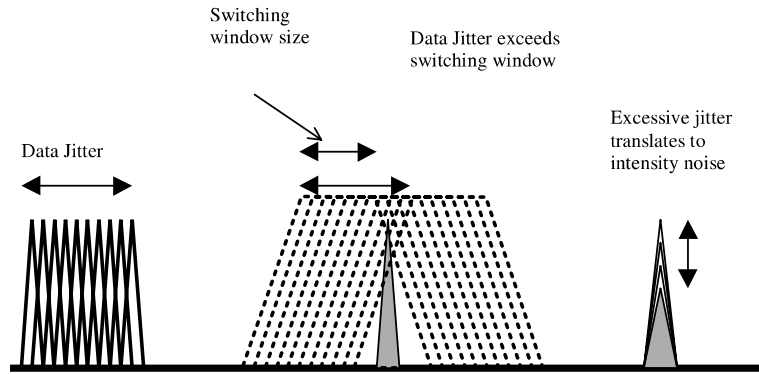


Fig. 8. Excess timing jitter can be converted to intensity noise.

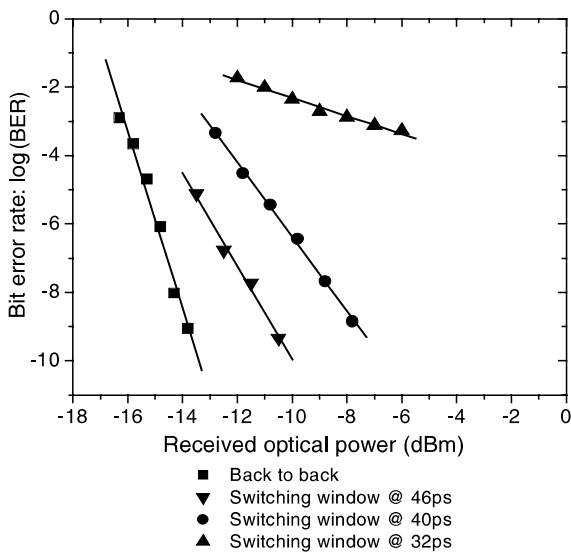


Fig. 9. Measured BER under different switching window size of TOAD.

#### 4. Discussions

In practical applications, timing jitter is a random process which has a significant influence on the system performance. Different systems will experience timing jitters with different amplitudes, frequency, and distribution functions. In our experiments, we use a low frequency speaker to add external timing jitter to the data signal, which provides a simple but efficient way to study the performance of an optical regenerator since the peak jitter value has the main influence on the re-

generated signal. In addition, by enhancing the SOA recovery rate of the TOAD through CW injection, the pattern dependent effect is suppressed. This allows the TOAD to regenerate data at 10 Gb/s and higher. Previous results published on the TOAD performances demonstrated that it could demultiplex OTDM data at speeds of 250 Gb/s and higher [5]. The difference between using the TOAD as a regenerator and as a demultiplexer is the entry point of the data. In demultiplexing, the high-speed multiplexed data is injected into the input port of the TOAD instead of the control port, and the data power at the input port is small in comparison with the saturation power of the SOA. The strong clock pulse enters the TOAD through the control port. However, as the clock pulse contains no data and therefore no associated pattern, pattern dependent effect does not pose a problem for demultiplexing. The entry point of the data is reversed for the case of optical regeneration. In optical regeneration, the data pulse is amplified and injected into the control port of the TOAD. Because of the higher power required to change the carrier population to provide the  $\pi$  phase shift, the input data gives rise to pattern dependent effects if the bit rate is comparable to the SOA recovery rate. Widening the TOAD window to accommodate the timing jitter requires that the SOA recover at an even faster rate than ordinary operating conditions.

In this paper, we analyzed the performance of the TOAD at 10 Gb/s with CW injection to enhance the SOA recovery rate. The performance of the TOAD at different timing jitter values as well

as different switching window sizes are measured using BER curves. Our results show that the TOAD can regenerate the signal at 10 Gb/s eliminating timing jitter of 17 ps, using a CW injection power of  $-0.9$  dBm.

### Acknowledgements

This work was supported by DARPA NGI program, F30602-00-2-0501. The authors would also like to thank G.K. Chang of Telcordia Technologies, Dan Brown and Alan Wilke of Multilink Technology for their support in the experiments.

### References

- [1] M. Eiselt, W. Pieper, H.G. Weber, SLALOM: semiconductor laser amplifier in a loop mirror, *Journal of Lightwave Technology* 13 (10) (1995) 2099–2111.
- [2] J.P. Sokoloff, P.R. Prucnal, I. Glesk, M. Kane, A terahertz optical asymmetric demultiplexer (TOAD), *Photonic Technology Letters* 5 (7) (1993) 787–790.
- [3] S. Fischer, M. Dulk, E. Gamper, W. Vogt, E. Gini, H. Melchoir, W. Hunziker, D. Nessel, A.D. Ellis, Optical 3R regenerator for 40 Gbit/s networks, *Electronics Letters* 35 (23) (1999) 2047–2049.
- [4] L. Billes, J.C. Simon, B. Kowalski, M. Henry, G. Michaud, P. Lamouler, F. Alard, 20 Gbit/s Optical 3R Regenerator using SOA based Mach–Zehnder Interferometer Gate, *ECOC 1997*, pp. 269–271.
- [5] I. Glesk, P.R. Prucnal, 250-Gb/s self clocked optical TDM with a polarization-multiplexed clock, *Fiber and Integrated Optics* 14 (1995) 71–82.
- [6] K.L. Deng, R.J. Runser, I. Glesk, P.R. Prucnal, Single-shot optical sampling oscilloscope for ultrafast optical waveforms, *Photonic Technology Letters* 10 (3) (1998) 397–399.
- [7] P. Toliver, I. Glesk, R.J. Runser, K.L. Deng, K.I. Kang, P.R. Prucnal, Ultrafast multihop packet-switched optical time-division multiplexing: components and systems, *Optical Engineering* 37 (12) (1998) 3187–3195.
- [8] A.J. Poustie, A.E. Kelly, R.J. Manning, K.J. Blow, All-optical regenerative memory with full write/read capability, *Optics Communications* 154 (1998) 277–281.
- [9] R.J. Manning, D.A.O. Davies, Three-wavelength device for all-optical signal processing, *Optics Letters* 19 (12) (1994) 889–891.
- [10] R.J. Manning, A.D. Ellis, A.J. Poustie, K.J. Blow, Semiconductor laser amplifiers for ultrafast all-optical signal processing, *Journal of the Optical Society of America B* 14 (11) (1997) 3204–3216.
- [11] A.D. Ellis, D.M. Patrick, D. Flannery, R.J. Manning, D.A.O. Davies, D.M. Spirit, Ultra-high-speed OTDM networks using semiconductor amplifier-based processing nodes, *Journal of Lightwave Technology* 13 (5) (1995) 761–770.
- [12] K. Inoue, M. Yoshino, Gain dynamics of a saturated semiconductor laser amplifier with  $1.47\text{ }\mu\text{m}$  LD pumping, *Photonic Technology Letters* 8 (4) (1996) 506–508.
- [13] M. Yoshino, K. Inoue, Improvement of saturation output power in a semiconductor laser amplifier through pumping light injection, *Photonic Technology Letters* 8 (1) (1996) 58–59.
- [14] P. Toliver, R.J. Runser, I. Glesk, P.R. Prucnal, Comparison of three nonlinear interferometric optical switch geometries, *Optics Communications* 175 (2000) 365–373.

## NEW GENERATION OF DEVICES FOR ALL-OPTICAL COMMUNICATIONS

**I. Glesk***Department of Optics, Faculty of Mathematics and Physics, Comenius University, Mlynská dolina, F-2, 842 48 Bratislava, Slovakia***R. J. Runser, P. R. Prucnal***Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA*

Received 18 September 2000, in final form 3 January 2001, accepted 4 January 2001

To increase the transmission capacity of future communication networks is becoming very critical. This task can only be accomplished by taking advantage of optical networks where multiplexing techniques such as Dense Wavelength Division Multiplexing (DWDM) and Optical Time Division Multiplexing (OTDM) are employed. To avoid electronic bottlenecks a whole new generation of ultrafast devices is needed. To fulfill these needs a new class of all optical devices has been proposed and developed. By taking advantage of the nonlinear dynamics in semiconductor optical amplifiers in combination with the fiber interferometers a new generation of ultrafast all-optical demultiplexers and wavelength converters has been demonstrated. Other switching technologies are also promising for the future. The latest technologies in the area of micro-machining have created very attractive low cost MEMS. Recently announced use of bubble technology for all-optical switching might also lead to the development of next generation large scale switching fabrics. This paper is an overview of the recent development in these areas.

PACS: 42.79.Sz, 42.79.Ta

**1 Introduction**

The explosive growth of the Internet has placed new demands on the bandwidth of the physical transport layer of the backbone network. While optical technology has begun satisfying the demand with high bandwidth dense wavelength division multiplexed (DWDM) point-to-point links, switching and routing packets has been performed using electronic hardware. Although electronics is sufficient for packet routing today, the tremendous growth in data traffic predicted over the next 5 years will push electronics to its fundamental limits. Current electronics that switch and route packets on the Internet rely upon integrated silicon (Si), gallium arsenide (GaAs), and indium phosphide (InP) devices. From a physics perspective, it is not likely that these technology will achieve terahertz speed needed for switching in the future Internet. New techniques are needed to alleviate the potential electronic bottleneck. It appears that optical technology will

be the only technology capable of achieving multi-terabit/second communications. For future generations of optical networks to utilize the full bandwidth of optical fiber, we expect data rates on each individual channel in DWDM networks to exceed the practical bit-rate of the driving electronics. To accommodate such high data rates, individual wavelength channels may be composed of modulated, picosecond mode-locked laser pulses from each data source. These new systems will optically aggregate traffic from many users into unique, closely spaced time slots to achieve extremely high data rates on each wavelength. By utilizing optical time division multiplexing (OTDM) technologies, the continued growth in capacity of fiber optic networks can be assured. Recent advances in OTDM have proven this technology's tremendous ability to perform high bandwidth switching among a large number of ports while offering aggregate capacities that exceed current electronic switched routers. Research groups throughout the world have begun exploring OTDM all-optical switching techniques.

## 2 All-optical devices

All-optical switches and demultiplexers are fundamental building blocks for enabling future OTDM systems [1]. Semiconductor optical nonlinearities with long recovery times ( $\geq 100$  ps) have been used to demonstrate efficient interferometric all-optical devices that promise to deliver switching and demultiplexing on terabit/s data streams. These nonlinearities are typically based upon a resonant excitation in actively-biased optical amplifiers or passive semiconductor nonlinear waveguides. Extensive experimental [2-7] and theoretical analysis [7-11] has been performed on various interferometric configurations of these devices. Due to their compact design, many of these switch architectures have been integrated, indicating their feasibility for future communication systems [12-17]. Optical nonlinearities in semiconductors are a very promising area for developing ultrafast and efficient optical switches [1]. Optical switches using actively-biased semiconductor optical amplifiers (SOAs) as the nonlinear switching element, have been used to demonstrate switching in systems using low control pulse energy (250 fJ) [18]. Although passive devices have demonstrated the shortest switching windows to date ( $\sim 200$  fs) [3], the passive bandfilling effect typically requires more optical control energy than actively-biased SOAs. Gain saturation induced bandfilling in active SOAs is enhanced by stimulated emission and therefore requires lower control pulse energy to generate sufficient nonlinearity for switching [16]. Also other sub-picosecond nonlinearities in semiconductors can be exploited to achieve ultrafast switching [1] and all-optical modulation [19].

### 2.1 Interferometric devices for all-optical processing

Interferometric devices for optical processing have been of great interest to the research community for some time [20] and gained momentum in the research community with the development of nonlinear optical loop mirrors (NOLMs) [21-23]. These devices, which simply consist of a 2x2 coupler and a long loop of fiber formed by joining the two fibers of one of the coupler's ends together, rely upon weak nonlinear interactions between a control and a signal pulse as they both co-propagate around the loop. If the nonlinear interaction is sufficiently large, a phase shift in the signal pulse propagating with the control pulse can be induced with respect to the counter-propagating signal pulse which does not travel with the control pulse. The change in phase alters the interference condition at the base of the loop when the signals recombine at the coupler and

switches the signal to the output port. Signals entering the loop in the absence of the control pulse, do not experience an appreciable phase change and are reflected back toward the source.

The switching windows for NOLMs can be made very short as they typically only depend upon the characteristics of the control pulse and the response of the nonlinearity. Indeed, switching experiments with temporal widths of 620 fs have been achieved [24]. However, NOLMs depend upon a weak nonlinear interaction in the fiber which usually requires high control pulse energies ( $>1$  pJ) and long fiber loop lengths ( $>100$  m) to generate a significant phase shift. Although there are many techniques to reduce both the control pulse energy requirements and loop lengths [23, 25], practical, compact devices for commercial optical communication systems have yet to be realized.

Finding a technique to reduce the control pulse energy and fiber lengths required in NOLMs relied upon using a nonlinear material other than fiber. Many groundbreaking experiments with semiconductor optical amplifiers (SOAs) inserted into the loop demonstrated that low energy optical pulses could change the gain of the amplifiers sufficiently to produce significant phase shifts in subsequent pulses passing through the amplifier [26]. As these integrated semiconductor amplifiers were very short ( $<1$  mm) they became a practical alternative to generating an optically induced nonlinearity. Additionally, the temporal onset of the phase shift was nearly as fast as the rising edge of the control pulse [27]. Unlike non-resonant fiber nonlinearity, however, this resonant, interband nonlinearity in the semiconductor material has a long relaxation time (100 to 500 ps). Efforts were soon underway to form a new class of switching devices based upon the efficient resonant nonlinearity in SOAs to induce a differential phase change between the two signal pulses counter-propagating in the fiber loop. The first device developed was known as a semiconductor laser amplifier in a loop mirror (SLALOM) and was used to investigate "contrast enhancement and optical correlation" [28]. Although the rising edge of the temporal switching window was a few picoseconds, the window's falling edge depended upon the gain recovery time of the SOA which was approximately 400 ps [28].

The last innovation to produce picosecond switching windows with SOAs was an architectural realization. It was discovered that the temporal duration of the window could be controlled by changing the asymmetric placement of the SOA. Due to the dynamics of this configuration, the switching window actually closes *earlier* than the recovery time of the SOA as the SOA is moved closer to the midpoint. Fig. 1 shows a schematic diagram of this device known as a Terahertz Optical Asymmetric Demultiplexer (TOAD) [2]. In the absence of a control pulse, data pulses enter the fiber loop, pass through the SOA at different times as they counter-propagate around the loop, and recombine interferometrically at the coupler. Since both pulses see the same medium as they propagate around the loop, the data is reflected back toward the source. In the presence of the control pulse, switching can occur. When a control pulse is injected into the loop, it saturates the SOA and changes its index of refraction. As a result, a differential phase shift can be achieved between the two counter-propagating data pulses to switch the data pulses to the output port. Only the pulses that co-propagate with and travel just behind the control pulse by up to twice the optical path length of the SOA offset are switched to the output port. All subsequent pulses will either see an unsaturated amplifier or a slowly recovering amplifier and will be reflected back toward the source. A polarization or wavelength filter is used at the output to reject the control and pass the switched data signal. The temporal duration of the switching window is determined by the offset of the SOA,  $\Delta x$ , from the center position of the loop. As this offset is reduced, the switching window size decreases. The size of the nominal switching

window duration,  $\tau_{win}$ , is related to the offset position by  $\tau_{win} = 2\Delta x / c_{fiber}$  (where  $c_{fiber}$  is the speed of light in fiber).

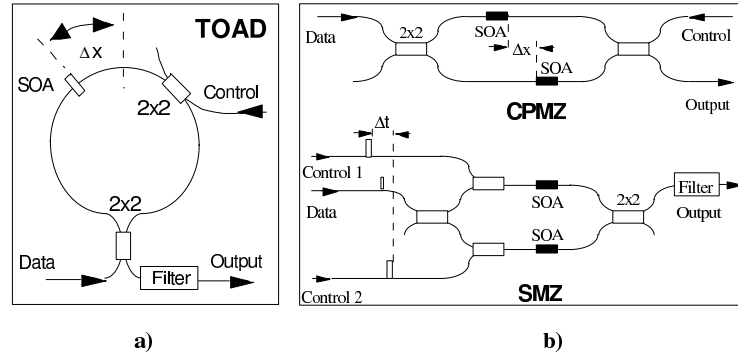


Fig. 1. Schematic diagram of ultrafast all-optical demultiplexers: a) The TOAD; b) CPMZ and SMZ.

By precisely controlling the offset position of the SOA, very short switching windows can be achieved. Demultiplexing of a single channel from a 250-Gb/s data stream has been demonstrated [29]. The practical control and data pulse energy requirements make it well-suited for typical communication signal powers. As the size of the device only depends upon the SOA length and offset from the center position in the loop, compact TOADs based upon discrete components have been constructed with loop lengths of less than 0.5 meter. The TOAD is robust to temperature variations and can be reliably operated without stabilization as data signals propagating in both directions around the loop experience the same effective medium. This device and its other variations may prove to be a practical approach to all-optical switching as they can be integrated using a variety of techniques that are discussed in this Section 2.2. The first experiments to evaluate the performance of the TOAD consisted of aggregating several pulses in time to form an ultrafast OTDM frame. Typically one pulse in the middle of the frame was modulated with a pseudorandom data pattern while the neighboring pulses were all set to 1. The demultiplexer was used to selected the modulated data signal from the ultrafast time frame. The first experiment demonstrated the TOAD's ability to switch pulses from a 50 Gb/s time frame to a baseband rate of 1.25 Gb/s [2]. This was subsequently followed by a 250 Gb/s demultiplexing experiment [29] and error-free demultiplexing from a continuous 160- Gb/s data stream [33]. Notable in all of these demonstrations was the low control pulse energy ( $< 1$  pJ) needed to perform the demultiplexing function.

Although the TOAD is based upon a Sagnac interferometer, other interferometric configurations are possible using a similar operating principle. These architectures improve the integratability and performance of the device although they may require active stabilization if constructed from discrete components. Two variations of the switch in a Mach-Zehnder interferometer configuration are shown in Fig. 1b. In the absence of the control signals, the Mach-Zehnder is balanced so that data signals are rejected at the output port. When control pulses are injected into the interferometer, a differential phase shift is briefly introduced between the two arms of the interferometer causing a data pulse to be switched to the output port. Similar to the TOAD,

subsequent data pulses that pass through the switch see the slow recovery of both SOAs and are rejected. The differences between the two Mach-Zehnder geometries shown is with respect to the propagation direction of control and data signals. In the Colliding Pulse Mach-Zehnder (CPMZ) shown in Fig. 1b, the data and control signals counter-propagate through the interferometer. As a result, a filter is not needed at the output to reject the control, and the control can be coupled into the interferometer without introducing additional coupling losses. The nominal switching window for the CPMZ is determined by the distance between the midpoints of the SOAs such that  $\tau_{win} = 2\Delta x / c_{fiber}$ . The other architecture known as the Symmetric Mach-Zehnder (SMZ) shown in Fig. 1b requires a filter at the output port to reject the control from the switched data signal since data and control signals co-propagate. Assuming the SOAs are positioned in the same relative location within the interferometer, the nominal switching window for the SMZ is determined by the temporal control pulse separation,  $\Delta t$ , of Control 1 and Control 2 prior to entering the interferometer such that  $\tau_{win} = \Delta t$ . Although the nominal switching window size provides an estimate of the switching window temporal duration, it does not account for the finite length of the SOAs. While the SOA length has little effect on the SMZ geometry, the minimum achievable switching windows for both the TOAD and CPMZ are constrained by the length of the SOAs [7, 8]. Many theoretical models have been developed to understand the ultrafast temporal response of SOAs [27, 30-32] and the characteristics of these optical switches [7-11].

With the successful development of all-optical demultiplexing, many new techniques have been used to enhance the performance of these devices. As the optical switching function is based upon gain saturation in an SOA, the repetition rate of the demultiplexing operation is somewhat limited by the recovery time of the amplifier. Novel optical biasing techniques using CW light have significantly reduced the recovery time [34]. It has been estimated that these techniques may enable the optical switch to function at repetition rates approaching 100 GHz [35]. Other demonstrations have shown that the TOAD can be successfully used to demultiplex many wavelengths simultaneously from an aggregated OTDM/WDM data stream [36].

*Gain-Transparent SOA-Switch* Dual wavelength operation of the TOAD/SLALOM configuration known as the Gain-Transparent SOA-Switch (*GT SOA-Switch*) has been proposed and demonstrated [37]. This device (Fig. 2) uses a data signal at a longer wavelength (1.55  $\mu\text{m}$ ) than the control signal (1.3  $\mu\text{m}$ ) so that it is far from the band edge of the optical amplifier. The technique enhances the signal-to-noise ratio of the device and can improve the switching contrast at the output. The GT SOA-switch has been successfully applied as an add/drop multiplexer [37] and to simultaneous demultiplexing of several wavelength channels from an OTDM/WDM data stream [38]. On other hand dual wavelength operation of such a switch could be too difficult to implement in the real optical network.

*Ultrafast Nonlinear Interferometer (UNI)*, developed at MIT Lincoln Labs, is another ultrafast all-optical OTDM switch using an SOA as the nonlinear element in a single-arm interferometer [4] (Fig. 3). By using a long length of Birefringent (PM) Fiber to separate orthogonally polarized components of data pulses in time, a control pulse can be introduced precisely between the components of a data pulse. When these components pass through the SOA, only the data pulse whose components are separated by the control pulse will experience a differential phase change. As a result, when the pulses are realigned by another long length of PM fiber, the components will interfere with each other. Only the pulse which experiences the differential phase change induced by the control pulse will be passed to the Output through the Polarization Filter (PM Filter). Although the TOAD/SLALOM and the UNI share several characteristics, the

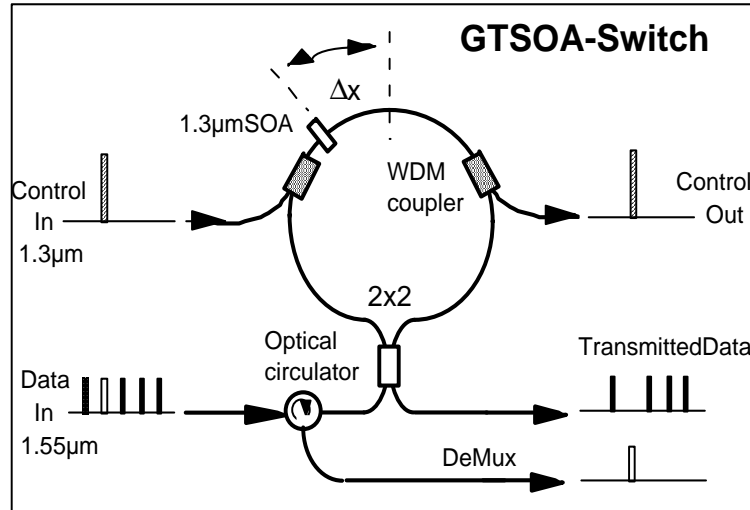


Fig. 2. GT SOA-Switch in Sagnac-interferometer TOAD/SLALOM configuration.

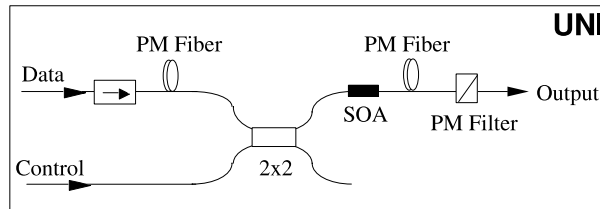


Fig. 3. Schematic diagram of Ultrafast Nonlinear Interferometer (UNI).

integratability and practicality of the UNI are limited by the long lengths of PM fiber needed to induce the polarization walk-off.

The switching window of the UNI is determined primarily by the birefringence of the PM fiber used to separate orthogonally polarized components of the data pulses in time. Enough walk-off is required to insert a control pulse between these two pulses. At a minimum, the walk-off should be longer than the control pulse width. Like any other SOA based switch, the UNI is limited by intraband carrier dynamics and carrier heating. Switching windows of about 1 ps can be expected. The switching repetition rate can be limited by the carrier recombination time in the SOA. However, 100 GHz repetition rates for bitwise logic functions have been reported [39]. As with any SOA-based switch, the UNI also has a noise background added to the switched signal due to spontaneous emission from the SOA. Noise figures in the range of 6 dB are typical for SOAs. Filtering and other techniques can be used to reduce the accumulation of noise in the signal for cascaded devices. Since the UNI requires at least 15 m of PM fiber to produce



the switching window [4], it is not likely that it will be easy to integrate. Since the UNI is dependent upon birefringence to achieve switching, the system must use extensive polarization control throughout the network to maintain reliability.

*Nonlinear Waveguide (NLWG) Switch.* Record breaking optical demultiplexing has recently been reported by a group at NEC Research working on a device known as a nonlinear waveguide (NLWG) switch. The group achieved 1.5 Tb/s demultiplexing with a 200 fs switching window at a repetition rate of 10 GHz [3]. The device uses a Mach-Zehnder interferometer configuration and passive semiconductor waveguides spatially offset within the arms to produce switching. This approach is very similar to the SMZ-TOAD configuration except that the semiconductor nonlinear waveguides are not actively biased. The NLWG switch uses data and control pulses sufficiently separated in wavelength to produce a bandfilling effect in the semiconductor waveguide. First, an interferometer is built using NLWGs in the arms (this may include a Sagnac, Mach-Zehnder, or even single-arm interferometer like the UNI). When a control pulse is introduced into the device at an appropriate wavelength, it is absorbed by the NLWG. The absorption creates an instantaneous refractive index change in the material through the bandfilling effect. Subsequently, data pulses which traverse the NLWG immediately after the control pulse can experience a differential phase change needed to produce switching and demultiplexing. The data and control pulses, which have different wavelengths, are separated at the output of the device using a bandpass filter. The switching window achieved in the most recent demonstration of the NLWG switch is the shortest to date ( $\sim 200$  fs). The small temporal window is a result of the nearly instantaneous index change of the semiconductor material from the control pulse. Unlike active SOA-based demultiplexers, the NLWG is a passive structure which does not exhibit intraband carrier dynamics or carrier heating. The nonlinear response can be almost as fast as the rising edge of the control pulse. The control pulse energy requirement of the NLWG device is one of its major limitations. Since the NLWG is passive, a significant amount of photons must be absorbed in the material to achieve an adequate phase shift for switching. For the InGaAsP waveguides at  $1.55 \mu\text{m}$ , a control pulse of nearly 5 pJ is required (after accounting fiber-to-chip coupling losses). To date, coupling efficiencies of only 10% have been achieved. As a result, a system built with NLWG demultiplexers would require control pulse energies of almost 50 pJ. Since the NLWG is passive, it is not likely that the control pulse energy can be reduced much beyond a few tens of picojoules. This greatly limits the device application to practical systems. Noise figure of a passive switch is typically not a problem and estimated to be less than 2 dB for these devices. Switching repetition rates of 40 GHz have been experimentally demonstrated by the NEC group [40].

## 2.2 Integration of All-Optical devices

We briefly review the progress that has been made in this area. While all of discussed devices presented can be constructed from discrete components, practical, high performance all-optical switches for commercial systems will most likely take advantage of photonic integration technology.

*Integrated all-optical switches.* The Sagnac, Mach-Zehnder, and Michelson interferometer all-optical switch geometries have been integrated by various groups. In order to fabricate these devices, both monolithic and hybrid technologies have been used. The first monolithically integrated nonlinear Sagnac interferometer capable of demultiplexing from 20 Gb/s to 10 Gb/s or 5

Gb/s was demonstrated by the Heinrich Hertz Institute (HHI) in 1996 [12]. Both the colliding pulse Mach-Zehnder (CPMZ) and symmetric Mach-Zehnder (SMZ) geometries have been integrated and subsequently demonstrated as high-speed demultiplexers by many groups [13-16]. Although the Mach-Zehnder configuration requires additional SOAs and couplers as compared to the Sagnac device, the Mach-Zehnder structures are more practical to fabricate since they do not require a large loop radius which may lead to bending losses in the waveguides. Furthermore, the SMZ has the inherent advantage that it exhibits the shortest switching window in the co-propagating configuration. Finally, an SOA-based optical switch using an integrated Michelson interferometer was used to demonstrate demultiplexing from 20 to 5 Gb/s [17]. The Michelson configuration may be a practical approach to integrated switching as anti-reflection coating is only applied to one side of the device and only two fiber-to-chip couplings are necessary for its operation as a demultiplexer [17].

The highest performance for demultiplexing has been demonstrated using the Mach-Zehnder structures. Both the CPMZ fabricated by HHI and the SMZ fabricated by Alcatel have been used to optically demultiplex from 40 to 10 Gb/s [13, 14]. The Alcatel monolithic SMZ is an all-active device as all waveguides contain an active SOA element fabricated on the same substrate. This device improves the optical power requirements by providing additional gain to account for fiber-to-chip coupling losses. A high performance monolithically integrated SMZ was also demonstrated by a collaboration among the Swiss Federal Institute of Technology, University of Denmark, and France Telecom. This group achieved reliable demultiplexing from 80 to 10 Gb/s [15]. The highest performance for an integrated SMZ to date has been achieved using a hybrid technique employed by NEC [16]. Fiber guides and passive silica waveguides are first fabricated onto a silicon planar lightwave circuit (PLC). The active SOA array chip is then flip chip mounted onto the PLC. Recently, NEC used this chip to demonstrate demultiplexing from 168 Gb/s to 10 Gb/s [16]. As integration technology in this area continues to mature, deployment of high performance optical switches in commercial systems becomes possible.

*Integrated all optical wavelength converter.* Wavelength conversion is one of the key functions which must be performed in existing DWDM optical networks. In current optical networks in order to perform conversion from wavelength  $\lambda_1$  to wavelength  $\lambda_2$ , an optical signal at  $\lambda_1$  must be first detected by a photoreceiver, then converted into an RF signal. This RF signal is now used to modulate a cw DFB laser to generate the required data at the new wavelength  $\lambda_2$ . This process is relatively slow and creates electronic bottlenecks in existing systems. This can be avoided if wavelength conversion is done all-optically for example using newly developed Integrated All-Optical Wavelength Converter 1901 ICM from Alcatel. Fig. 3 is a schematic diagram of such a device. This converter exploits cross-phase modulation in an integrated Mach-Zehnder (MZ) interferometer based on an all-active MZ-SOA structure. An input modulated signal at a wavelength  $\lambda_1$  modulates the carrier density in the SOA inside of the interferometer, producing a modulation of its refractive index. This in turn leads to phase modulation of an injected CW beam at the desired output wavelength  $\lambda_2$ , which is converted to amplitude modulation via the MZ interferometer. The signal data pattern is therefore transferred to the new wavelength  $\lambda_2$ .

The nonlinear transfer function of the device allows both enhancement of the signal extinction ratio and compression of the optical noise amplitude.

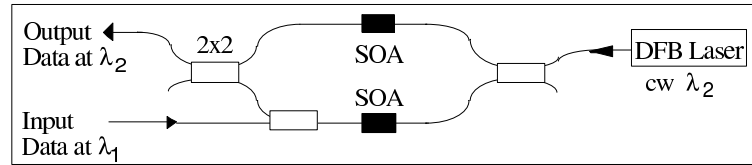


Fig. 4. All-Optical Interferometric Wavelength Converter - schematic diagram.

### 3 Opto-mechanical switching devices

While the previous analysis in Chapter 2 only considered interferometric based all-optical switches, other mechanisms for optical switching are also being pursued vigorously. The most common optical switching fabric that is currently being integrated into commercial packet switching systems is based upon micro-electro-mechanical systems or MEMS. Simple embodiments of the MEMS technology include movable micro-mirrors that route beams of light according to their destination. Early this year Agilent Technologies gained a lot of attention by announcing its capability to use a bubble technology for optical switching and routing. These two promising technologies are discussed in this section.

#### 3.1 Movable Mirror Architectures

One of the leader in MEMS technology today is Lucent Technologies. Lucent is poised to offer their first all-optical routing system this year called the WaveStar LambdaRouter. The LambdaRouter uses a 256x256 array of movable mirrors to direct light from one fiber to another. Advantages of the MEMS architecture include scalability, low power consumption, low loss, compact size, and protocol transparency. Lucent's current system can support single channel data rates as high as 40 Gb/s. MEMS offers a simple solution to the optical switching problem and avoids the electronic conversion required in standard routers but the applications area is somewhat limited. Since MEMS are inherently mechanical, they are limited in speed. The Lucent LambdaRouter can move its mirrors only on a time scale of 10 ms. While this is appropriate for optical circuit switching and optical layer restoration protection switching, it is not nearly fast enough to support switching on a packet-by-packet basis required by IP routing. Furthermore electronic hardware must still be used to obtain the routing information to control the switch. Due to the mechanical nature of MEMS, long-term reliability and packaging are still critical issues in these systems that will be proved over time. Additional advances in MEMS will most likely be able to upgrade the speed of these switches. The MEMS based switches will most likely interconnect service providers and large cities where continuous traffic streams are established for longer periods of time between fixed locations.

#### 3.2 Bubble Technology

Agilent Technologies has been a pioneer of ink-jet technology for low-cost color printers. This same technology has now been applied to an all-optical switch fabric and commercial systems

are expected by the end of year 2000. The current prototype demonstration is a 32x32 all-optical switching matrix. These new photonic switches are based on technology that uses a combination of reliable inkjet and planar lightwave circuit technologies. They accomplish the task of re-directing light without the help of mirrors or any other moving parts. This switch is composed of a vertical and horizontal array of permanently aligned waveguides. Light is transmitted across a horizontal path from the input to output port until a switch command is issued. When commanded, a bubble is created at the intersection of appropriate waveguides and the light is reflected down a vertical path to the switched port. This bubble is formed using the same reliable technology now used in inkjet printers. Like MEMS, bubble technology has low loss, format and protocol transparency, and compact size. The bubble technology may prove to be more reliable than MEMS based switches since there is less moving parts involved in the switching operation. Little information is available about the capabilities of this new, proprietary technology. It is estimated based upon the ink-jet printing technology, that switching latency will fall in the millisecond range. The Agilent switch might find applications in optical circuit switching and optical layer restoration protection switching.

#### 4 Conclusions

Although electronics has made great strides toward satisfying the switching bandwidth of future communication networks, it does not appear that electronic switches will reach the targeted terabit/second regime even with the highest degree of parallelism. However, based on presented results these newly developing technologies might provide a revolutionary breakthrough in scalability, bandwidth, reliability, and speed. In conclusion, Table I summarizes the key device parameters of the technologies presented in this paper.

Table I. Comparison of four types of optical switches.

Device	Switching Time	Repetition Rate	Control Pulse Energy (pJ)	Noise Figure (dB)	Integrat.
TOAD/SLALOM	< 1 ps	100+ GHz	0.25	6	YES
UNI	< 1 ps	100+ GHz	0.25	6	NO
NLWG	0.2 ps	40+ GHz	50	low	YES
NOLM	0.8 ps	100+ GHz	50+	low	NO
MEMS	10 ms	< 1 kHz	N/A	N/A	YES
Bubble	10 ms	< 1 kHz	N/A	N/A	YES

#### References

- [1] O. Wada: *Opt. Quant. Electron.* **32** (2000) 453
- [2] J. P. Sokoloff, P. R. Prucnal, I. Glesk, M. Kane: *A Terahertz Optical Asymmetric Demultiplexer (TOAD)*, *OSA Proceedings on Photonics in Switching*, (Eds. J. W. Goodman, R. C. Alferness) Optical Society of America, Washington, D.C. 1993 **16**, PD-4; J. P. Sokoloff, P. R. Prucnal, I. Glesk, M. Kane: *IEEE Photon. Technol. Lett.* **5** (1993) 787

- [3] S. Nakamura, Y. Ueno, K. Tajima: *IEEE Photon. Technol. Lett.* **10** (1998) 1575
- [4] N. S. Patel, K. L. Hall, K. A. Rauschenbach: *Optics Lett.* **21** (1996) 1466
- [5] A. D. Ellis, D. M. Patrick, D. Flannery, R. J. Manning, D. A. O. Davies, D. M. Spirit: *J. Lightwave Technol.* **13** (1995) 761
- [6] M. Eiselt, W. Pieper, H. G. Weber: *Electron. Lett.* **29** (1993) 1167
- [7] P. Toliver, R. J. Runser, I. Glesk, P. R. Prucnal: *Optics Comm.* **175** (2000) 365
- [8] K. I. Kang, T. G. Chang, I. Glesk, P. R. Prucnal: *Appl. Opt.* **35** (1996) 417
- [9] R. J. Manning, A. D. Ellis, A. J. Poustie, K. J. Blow: *J. Opt. Soc. Am. B* **14** (1997) 3204
- [10] N. S. Patel, K. L. Hall, K. A. Rauschenbach: *Appl. Optics* **37** (1998) 2831
- [11] S. Nakamura, K. Tajima: *Jpn. J. Appl. Phys.* **35** (1996) L1426
- [12] E. Jahn, N. Agrawal, W. Pieper, H.-J. Ehrke, D. Franke, W. Furst, C. M. Weinert: *Electron. Lett.* **32** (1996) 782
- [13] E. Jahn, N. Agrawal, M. Arbert, H.-J. Ehrke, D. Franke: *Electron. Lett.* **31** (1995) 1857
- [14] D. Wolfson, A. Kloch, T. Fjelde, C. Janz, B. Dagens, M. Renaud: *IEEE Photon. Technol. Lett.* **12** (2000) 332
- [15] R. Hess, M. Caraccia-Gross, W. Vogt, E. Gamper, P. A. Besse, M. Duelk, E. Gini, H. Melchior, B. Mikkelsen, M. Vaa, K. S. Jepsen, K. E. Stubkjaer, S. Bouchoule: *IEEE Photon. Technol. Lett.* **10** (1998) 165
- [16] K. Tajima, S. Nakamura, Y. Ueno, J. Sasaki, T. Sugimoto, T. Kato, T. Shimoda, H. Hatakeyama, T. Tamanuki, T. Sasaki: *IEICE Trans. on Electron. E* **83C** (2000) 959
- [17] B. Mikkelsen, M. Vaa, N. Storkfelt, T. Durhuus, C. Joergensen, R. J. S. Pedersen, S. L. Danielsen, K. E. Stubkjaer, M. Gustavsson, W. van Berlo: *Monolithic integrated Michelson interferometer with SOAs for high-speed all-optical signal processing*, *Proc. OFC'95* San Diego, CA 1995, pp. 13-14, paper TuD4
- [18] K.-L. Deng, R. J. Runser, P. Toliver, C. Coldwell, D. Zhou, I. Glesk, P. R. Prucnal: *Electron. Lett.* **34** (1998) 2418
- [19] Neogi, O. Wada, Y. Takahashi, H. Kawaguchi: *Opt. Lett.* **23** (1998) 1212
- [20] K. Otsuka: *Opt. Lett.* **8** (1983) 471
- [21] N. J. Doran, D. Wood: *Opt. Lett.* **13** (1988) 56
- [22] N. J. Doran, D. S. Forrester, B. K. Nayar: *Electron. Lett.* **25** (1989) 267
- [23] M. Jinno, T. Matsumoto: *Opt. Lett.* **16** (1991) 220
- [24] E. Yamada, K. Suzuki, M. Nakazawa: *Electron. Lett.* **30** (1994) 1966
- [25] M. Asobe, T. Ohara, I. Yokohama, T. Kaino: *Electron. Lett.* **32** (1996) 1396
- [26] A. W. O'Neill, R. P. Webb: *Electron. Lett.* **26** (1990) 2008
- [27] J. Mark, J. Mork: *Appl. Phys. Lett.* **61** (1992) 2281
- [28] M. Eiselt: *Electron. Lett.* **28** (1992) 1505
- [29] I. Glesk, J. P. Sokoloff, P. R. Prucnal: *Electron. Lett.* **30** (1994) 339
- [30] K. L. Hall, G. Lenz, A. M. Darwish, E. P. Ippen: *Opt. Comm.* **111** (1994) 589
- [31] J. M. Tang, K. A. Shore: *IEEE J. Quantum Electron.* **34** (1998) 1263
- [32] M. Y. Hong, Y. H. Chang, A. Dienes, J. P. Heritage, P. J. Delfyett: *IEEE J. Quantum Electron.* **30** (1994) 1122
- [33] K. Suzuki, K. Iwatsuki, S. Nishi, M. Saruwatari: *Electron. Lett.* **30** (1994) 1501
- [34] R. J. Manning, D. A. O. Davies: *Opt. Lett.* **19** (1994) 889

- [35] R. J. Manning, D. A. O. Davies, D. Cotter, J. K. Lucek: *Electron. Lett.* **30** (1994) 787
- [36] B. K. Mathason, H. Shi, I. Nitta, J. Abeles, J. C. Connolly, P. J. Delfyett: *IEEE Photon. Technol. Lett.* **11** (1999) 331
- [37] S. Diez, R. Ludwig, H. G. Weber: *IEEE Photon. Technol. Lett.* **11** (1999) 60
- [38] S. Diez, R. Ludwig, H. G. Weber: *Electron. Lett.* **34** (1998) 803
- [39] K. L. Hall, K. A. Rauschenbach: *Opt. Lett.* **23** (1998) 1271
- [40] K. Tajima, S. Nakamura, Y. Ueno: *Mat. Sci. Eng. B - Solid B* **48** (1997) 88



*Invited paper*

## Interferometric ultrafast SOA-based optical switches: From devices to applications

ROBERT J. RUNSER<sup>1\*</sup>, DEYU ZHOU<sup>2</sup>, CHRISTINE COLDWELL<sup>3</sup>,  
BING C. WANG<sup>1</sup>, PAUL TOLIVER<sup>2</sup>, KUNG-LI DENG<sup>4</sup>, IVAN GLESK<sup>1</sup>  
AND PAUL R. PRUCNAL<sup>1</sup>

<sup>1</sup>*Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA*

<sup>2</sup>*Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA*  
(Present address: Telcordia Technologies, Red Bank, NJ 07701, USA)

<sup>3</sup>*Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA*  
(Present address: Ciena Corporation, Linthicum, MD 21090, USA)

<sup>4</sup>*Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA*  
(Present address: Jedai Broadband Networks, Eatontown, NJ, USA)

(\*author for correspondence: E-mail: rrunser@ee.princeton.edu)

Received 16 August 2000; accepted 1 November 2000

**Abstract.** All-optical switches are fundamental building blocks for future, high-speed optical networks that utilize optical time division multiplexing (OTDM) techniques to achieve single channel data rates exceeding 100 Gb/s. Interferometric optical switches using semiconductor optical amplifier (SOA) nonlinearities perform efficient optical switching with  $< 500$  fJ of control energy and are approaching optical sampling bandwidths of nearly 1 THz. In this paper, we review work underway at Princeton University to characterize and demonstrate these optical switches as processing elements in practical networks and systems. Three interferometric optical switch geometries are presented and characterized. We discuss limitations on the minimum temporal width of the switching window and prospects for integrating the devices. Using these optical switches as demultiplexers, we demonstrate two 100-Gb/s testbeds for photonic packet switching. In addition to the optical networking applications, we have explored simultaneous wavelength conversion and pulse width management. We have also designed high bandwidth sampling systems using SOA-based optical switches as analog optical sampling gates capable of analyzing optical waveforms with bandwidths exceeding 100 GHz. We believe these devices represent a versatile approach to all-optical processing as a variety of applications can be performed without significantly changing the device architecture.

**Key words:** optical sampling, optical switching, optical time division multiplexing (OTDM), photonic networks, semiconductor optical amplifier (SOA), terahertz optical asymmetric demultiplexer (TOAD)

### 1. Introduction

Fiber optic communication networks have continued to increase in capacity largely due to advances in the research and development of the optical components and subsystems that constitute the physical layer. The most successful technologies that continue to improve the performance of these networks have taken advantage of component miniaturization and integration while

remaining cost competitive. As new technologies become available, the higher layers of the network have successfully adapted to accommodate the performance strengths of new systems. For future generations of optical networks to utilize the full bandwidth of optical fiber, we expect data rates on each individual channel in dense wavelength division multiplexed (DWDM) networks to exceed the practical bit-rate of the driving electronics. To accommodate such high data rates, individual wavelength channels may be composed of modulated, picosecond mode-locked laser pulses from each data source. These new systems will optically aggregate traffic from many users into unique, closely spaced time slots to achieve extremely high data rates on each wavelength. By utilizing optical time division multiplexing (OTDM) technologies, the continued growth in capacity of transmission links and scalability of switching fabrics in fiber optic networks can be assured.

All-optical switches and demultiplexers are fundamental building blocks for enabling future optical TDM systems (Wada 2000). Semiconductor optical nonlinearities with long recovery times ( $> 100$  ps) have been used to demonstrate efficient interferometric all-optical switches that promise to deliver switching and demultiplexing on terabit/s data streams. These nonlinearities are typically based upon a resonant excitation in actively biased optical amplifiers or passive semiconductor nonlinear waveguides. Extensive experimental (Eiselt *et al.* 1993; Sokoloff *et al.* 1993; Ellis *et al.* 1995; Patel *et al.* 1996; Nakamura *et al.* 1998; Toliver *et al.* 2000a) and theoretical analyses (Kang *et al.* 1996a; Nakamura and Tajima 1996; Manning *et al.* 1997; Patel *et al.* 1998; Toliver *et al.* 2000a) have been performed on various interferometric configurations of these devices. Due to their compact design, many of these switch architectures have been integrated, indicating their feasibility for future communication systems (Jahn *et al.* 1995; Mikkelsen *et al.* 1995; Jahn *et al.* 1996; Hess *et al.* 1998; Tajima *et al.* 2000; Wolfson *et al.* 2000).

Optical nonlinearities in semiconductors are a promising area for developing ultrafast and efficient optical switches (Wada 2000). Optical switches using actively biased semiconductor optical amplifiers (SOAs) as the nonlinear switching element, have demonstrated switching in systems using low control pulse energy (250 fJ) (Deng *et al.* 1998a). Although passive devices have demonstrated the shortest switching windows to date ( $\sim 200$  fs) (Nakamura *et al.* 1998), the passive bandfilling effect (BFE) typically requires more optical control energy than actively biased SOAs. Gain saturation induced bandfilling in active SOAs is enhanced by stimulated emission and therefore requires lower control pulse energy to generate sufficient nonlinearity for switching (Tajima *et al.* 2000). Although other sub-picosecond nonlinearities in semiconductors can be exploited to achieve ultrafast switching (Wada 2000) and all-optical modulation (Neogi *et al.* 1998), we focus our attention in this paper on the devices using gain saturation induced nonlinearity in SOAs. The practical energy requirements of actively biased



SOA-based optical switches indicate that these devices may be implemented in future commercial systems where all-optical processing is required.

A unique advantage of these optical switching structures is their applicability to a variety of different applications. In addition to offering increased capacity in lightwave networks, these devices have also been successfully used as all-optical sampling gates for optical test and measurement applications. New systems that require all-optical processing can be more easily realized as many applications can be accomplished by simply changing the configuration of the device.

In this paper, we review many of the most promising applications of ultrafast, SOA-based optical switches under investigation at Princeton University. We introduce the architectures for the all-optical switches and characterize their performance in Section 2. Section 3 provides a review of the ultrafast 100-Gb/s optical TDM network testbeds that have been demonstrated using all-optical switching technology. In addition to high-speed optical header recognition and demultiplexing, new optical processing applications are possible by varying the temporal width of the optical switching window. Section 4 introduces a novel  $1 \times 2$  all-optical routing switch for high-speed optical packets using wide switching windows. Variation of the switching window width creates a new, flexible technique for simultaneous all-optical format and wavelength conversion, which is described in Section 5. This technique is a key enabling technology to the design of a transparent optical interface between WDM and optical TDM networks. Section 6 presents two analog optical sampling systems employing all-optical sampling gates that can be used for optical diagnostics and communication measurement applications. Finally, a brief discussion of the remaining technical challenges and future prospects of interferometric SOA-based optical switches is presented in the concluding section.

## 2. Development of ultrafast SOA-based optical switches

The fundamental principles and architectures for interferometric SOA-based optical switches are described in this section. We begin with a brief historical introduction followed by experimentally measured characteristics of these devices. Finally, we review recent efforts underway to integrate optical switches using SOAs.

### 2.1. HISTORICAL DEVELOPMENT

Interferometric devices for optical processing and logic have been of great interest to the research community for some time (Otsuka 1983). Optical

switching and demultiplexing gained momentum in the research community with the development of nonlinear optical loop mirrors (NOLMs) (Doran and Wood 1988; Doran *et al.* 1989; Jinno and Matsumoto 1991). These devices, which simply consist of a  $2 \times 2$  coupler and a long loop of fiber formed by joining the two fibers of one of the coupler's ends together, rely upon weak nonlinear interactions between a control and a signal pulse as they both co-propagate around the loop. If the nonlinear interaction is sufficiently large, a phase shift in the signal pulse propagating with the control pulse can be induced with respect to the counter-propagating signal pulse that does not travel with the control pulse. The change in phase alters the interference condition at the base of the loop when the signals recombine at the coupler and switches the signal to the output port. Signals entering the loop in the absence of the control pulse, do not experience an appreciable phase change and are reflected back toward the source.

The switching windows for NOLMs can be made very short as they typically only depend upon the characteristics of the control pulse and the response of the nonlinearity. Indeed, switching experiments with temporal widths of 620 fs have been achieved (Yamada *et al.* 1994). However, NOLMs depend upon a weak nonlinear interaction in the fiber which usually requires high control pulse energies ( $>1$  pJ) and long fiber loop lengths ( $>100$  m) to generate a significant phase shift. Although there are many techniques to reduce both the control pulse energy requirements and loop lengths (Yamada *et al.* 1994; Asobe *et al.* 1996), practical, NOLMs for commercial optical communication systems have yet to be realized.

Finding a technique to reduce the control pulse energy and fiber lengths required in NOLMs relied upon using a nonlinear material other than fiber. Many groundbreaking experiments with SOAs inserted into the loop demonstrated that low energy optical pulses could change the gain of the amplifiers sufficiently to produce significant phase shifts in subsequent pulses passing through the amplifier (O'Neill and Webb 1990). As these integrated semiconductor amplifiers were very short ( $<1$  mm) they became a practical alternative to generating an optically induced nonlinearity. Additionally, the temporal onset of the phase shift was nearly as fast as the rising edge of the control pulse (Mark and Mork 1992). Unlike non-resonant fiber nonlinearity, however, this resonant, interband nonlinearity in the semiconductor material has a long relaxation time (100–500 ps). Efforts were soon underway to form a new class of switching devices using the efficient nonlinearity in SOAs to induce a differential phase change between the two signal pulses counter-propagating in the fiber loop. The first device developed was known as a semiconductor laser amplifier in a loop mirror (SLALOM) and was used to investigate "contrast enhancement and optical correlation" (Eiselt 1992). Although the rising edge of the temporal switching window was a few picoseconds, the window's falling edge

depended upon the gain recovery time of the SOA which was approximately 400 ps (Eiselt 1992).

The last innovation to produce picosecond switching windows with SOAs was an architectural realization. It was discovered that the temporal width of the switching window could be linearly controlled by adjusting the displacement of the SOA from the midpoint of the loop. Due to the dynamics of this configuration, the switching window actually closes *earlier* than the recovery time of the SOA as the SOA is moved closer to the midpoint. Fig. 1 shows a schematic diagram of this device known as a terahertz optical asymmetric demultiplexer (TOAD) (Sokoloff *et al.* 1993). In the absence of a control pulse, data pulses enter the fiber loop, pass through the SOA at different times as they counter-propagate around the loop, and recombine interferometrically at the coupler. Since both pulses experience the same effective medium as they propagate around the loop, the data is reflected back toward the source. In the presence of the control pulse, switching can occur. When a control pulse is injected into the loop, it saturates the SOA and changes its index of refraction. As a result, a differential phase shift can be achieved between the two counter-propagating data pulses to switch the data pulses to the output port. Only the pulses that co-propagate with and follow just behind the control pulse by up to twice the optical path length of the SOA offset are switched to the output port. All subsequent data pulses will either see an unsaturated amplifier or a slowly recovering amplifier and will be reflected back toward the source. A polarization or wavelength filter is used at the output to discriminate the switched data signal from the control pulse. The temporal duration of the switching window is determined by the offset of the SOA,  $\Delta x_{\text{SOA}}$ , from the center position of the loop. As this offset is reduced, the switching window size decreases. The size of the nominal switching window duration,  $\tau_{\text{win}}$ , is related to the offset position by  $\tau_{\text{win}} = 2\Delta x_{\text{SOA}}/c_{\text{fiber}}$  (where  $c_{\text{fiber}}$  is the speed of light in fiber).

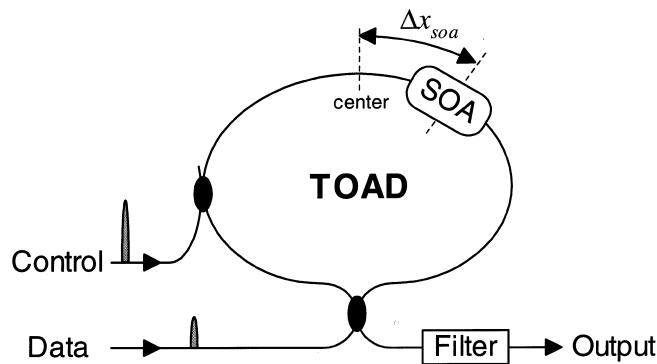


Fig. 1. Schematic of a TOAD. The displacement of the SOA from the center of the loop determines the size of the switching window. A filter at the output rejects the control pulse from the switched data signal.

By precisely controlling the offset position of the SOA, very short switching windows can be achieved. Demultiplexing of a single channel from a 250-Gb/s data stream has been demonstrated (Glesk *et al.* 1994). As the size of the device only depends upon the SOA length and offset from the center position in the loop, compact TOADs based upon discrete components have been constructed with loop lengths of less than 1 m.

The TOAD is an ideal candidate for practical demultiplexing in optical communication networks due to its low control energy. To achieve full switching, the differential phase shift between the counter-propagating data pulses should approach  $\pi$ . For a 500  $\mu\text{m}$  long InGaAsP SOA ( $n_{\text{SOA}} = 3.3$ ) at 1.55  $\mu\text{m}$ , the index change is on the order of  $10^{-3}$ . It has been estimated that the carrier density change associated with this phase shift is on the order of  $10^{17} \text{ cm}^{-3}$  (Cotter *et al.* 1999) which is approximately a 10% change in the carrier density of an appropriately biased SOA in population inversion. Both theoretical calculations (Kang *et al.* 1996b; Manning *et al.* 1997) and experimental measurements (Kang *et al.* 1996c) have verified that a control pulse with 10% of the SOA saturation energy is sufficient to induce a  $\pi$  phase shift. Depending upon the SOA parameters, current bias, and wavelength, control pulse energies of  $< 500 \text{ fJ}$  can typically be used. The control pulse energy can fluctuate by nearly 20% without significantly degrading the signal-to-noise ratio (SNR) at the switched output (Zhou *et al.* 1999).

Amplified spontaneous emission (ASE) noise, data pulse saturation, and component tolerances can also impact the TOAD output SNR (Zhuo *et al.* 1999). As compared to other forms of optical amplification, SOAs have relatively high noise figures approaching 6–8 dB. By choosing the proper data pulse energy and employing an optical bandpass filter, error free operation of the demultiplexer has been achieved in the presence of this noise (Deng *et al.* 1998a). Other techniques such as dual wavelength operation of the optical switch can nearly eliminate the influence of ASE on the switched output signal (Diez *et al.* 1999a). Although sufficient data pulse energy is needed for detection above the background noise, the maximum allowable data energy is constrained by the saturation characteristics of the SOA. Theoretical and experimental studies of the data signal saturation effect constrain the data energy to approximately no more than 5–10% of the control signal (Kang *et al.* 1996b). This rule, however, depends heavily upon the optical TDM data bit rate as well as the TOAD operating characteristics.

The TOAD is robust to temperature variations and can be reliably operated without stabilization as data signals propagating in both directions around the loop experience the same effective medium. The architecture can also be adapted to accommodate other short, highly nonlinear materials in place of the SOA to operate the device with different wavelengths or to achieve performance enhancements. This device and its other variations may prove to be

a practical approach to all-optical switching as they can be integrated using a variety of techniques that are discussed later in this section.

## 2.2. OTHER OPTICAL SWITCH GEOMETRIES

Although the TOAD is based upon a Sagnac interferometer, other interferometric configurations are possible using a similar operating principle. These architectures improve the integratability and performance of the device, although they may require active stabilization if constructed from discrete components. Two variations of the switch in a Mach-Zehnder interferometer configuration are shown in Fig. 2. In the absence of the control signals, the Mach-Zehnder is balanced so that data signals are rejected from the output port. When control pulses are injected into the interferometer, a differential phase shift is briefly introduced between the two arms of the interferometer causing a data pulse to be switched to the output port. Similar to the TOAD, subsequent data pulses that pass through the switch see the slow recovery of both SOAs and are rejected. The difference between the two Mach-Zehnder geometries shown is with respect to the propagation direction of control and data signals. In the colliding pulse Mach-Zehnder (CPMZ) shown in Fig. 2(a), the data and control signals counter-propagate through the interferometer. As a result, a filter is not needed at the output to reject the control, and the control can be coupled into the interferometer without introducing additional coupling losses. The nominal switching window for the CPMZ is determined by the distance between the midpoints of the SOAs such that  $\tau_{\text{win}} = 2\Delta x_{\text{SOA}}/c_{\text{fiber}}$ . The other architecture known as the symmetric Mach-Zehnder (SMZ), shown in Fig. 2(b), requires a filter at the output port to reject the control from the

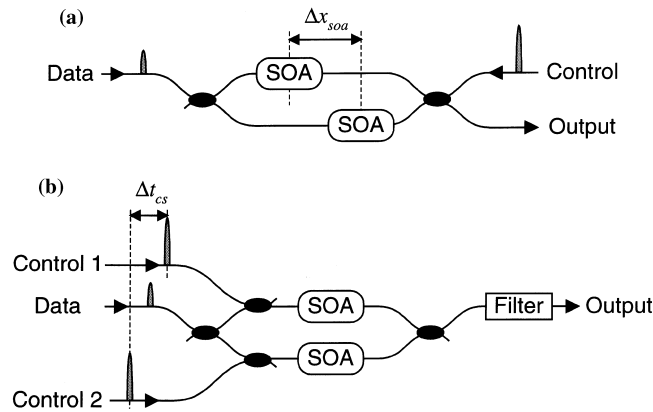


Fig. 2. Two Mach-Zehnder optical switch geometries: (a) the colliding pulse Mach-Zehnder (CPMZ) and (b) the symmetric Mach-Zehnder (SMZ).

switched data signal since data and control signals co-propagate. Assuming the SOAs are positioned in the same relative location within the interferometer, the nominal switching window for the SMZ is determined by the temporal control pulse separation,  $\Delta t_{cs}$ , of Control 1 and Control 2 prior to entering the interferometer such that  $\tau_{win} = \Delta t_{cs}$ .

Although the nominal switching window size provides an estimate of the switching window temporal duration, it does not account for the finite length of the SOAs. While the SOA length has little effect on the SMZ geometry, the minimum achievable switching windows for both the TOAD and CPMZ are constrained by the length of the SOAs (Kang *et al.* 1996a; Toliver *et al.* 2000a).

### 2.3. EXPERIMENTAL EVALUATION OF OPTICAL SWITCH PERFORMANCE

While many theoretical models have been developed to understand the ultrafast temporal response of SOAs (Mark and Mork 1992; Hall *et al.* 1994; Hong *et al.* 1994; Tang and Shore 1998) and the characteristics of these optical switches (Kang *et al.* 1996a; Nakamura and Tajima 1996; Manning *et al.* 1997; Patel *et al.* 1998; Toliver *et al.* 2000a), we present a summary of the experimentally measured switching performance of the three geometries introduced in the previous section. All three switches were constructed from similar discrete components. The switching windows were achieved by using a scanning pump-probe apparatus that is described in Toliver *et al.* (2000a). The apparatus is used to vary the delay path of the data pulses in time over a 40-ps range with respect to the control pulses. The output of the system is a convolution of the data pulses with the transfer function of the switching window induced by the control. This technique provides a means of rapidly characterizing the switching window. While the TOAD is based upon the inherently stable Sagnac interferometer, thermal variations in the optical fiber cause the output of the Mach-Zehnder switches to fluctuate slowly in time. By performing the scan at a rate faster than the thermal variations, switching windows of the fiber-based Mach-Zehnder geometries can be obtained without resorting to complex stabilization techniques. (Note that thermal variations do not significantly affect the stability of any of these interferometers if integrated devices with short optical path lengths are used.)

The switching window provides information regarding the shape, amplitude, and temporal width of the optical transfer function. This characterization is instrumental in determining the optical demultiplexing and sampling bandwidth of the switch. For the results described here, various pulse energies, widths, and repetition rates were used to demonstrate the effect these parameters have on the switching performance. For the TOAD windows shown in Fig. 3(a), the data and control pulses were set to energies of 5 and

20 fJ respectively. An erbium-doped fiber modelocked laser produced 2.3-ps pulses at a 10-GHz repetition rate. The long rising edge of the switching window is due to the finite length of the SOA whereas the sharp falling edge is only limited by the data and control pulse widths. The results indicate that the switching window decreases nearly linearly as the SOA offset position within the loop is decreased. As the switching window becomes shorter, the amplitude also is reduced once the offset becomes less than the length of the SOA. As a portion of the SOA straddles the midpoint of the loop, the available nonlinearity is reduced and less contrast is observed at the output. Additionally, at extremely small offsets, the switching window width does not decrease further since the finite data and control pulse widths become the dominant limitation. This trend continues until the effective switching offset is 0 ps at which point the switching window nearly vanishes. The shortest switching window achieved with this experiment was about 3.8 ps at FWHM.

The CPMZ switch was also investigated under similar circumstances. In this case the data and control pulses were approximately 1.6 ps in width with similar pulse energies as before. The results of these scans are shown in Fig. 3(b). In contrast to the TOAD, the counter-propagating geometry significantly increases both the rise and fall times of the window edges. This is primarily a function of the finite SOA lengths, which were each approximately 500  $\mu\text{m}$ . The architecture of the CPMZ limits its performance to minimum switching window widths of about 8 ps for this SOA length (Toliver *et al.* 2000a).

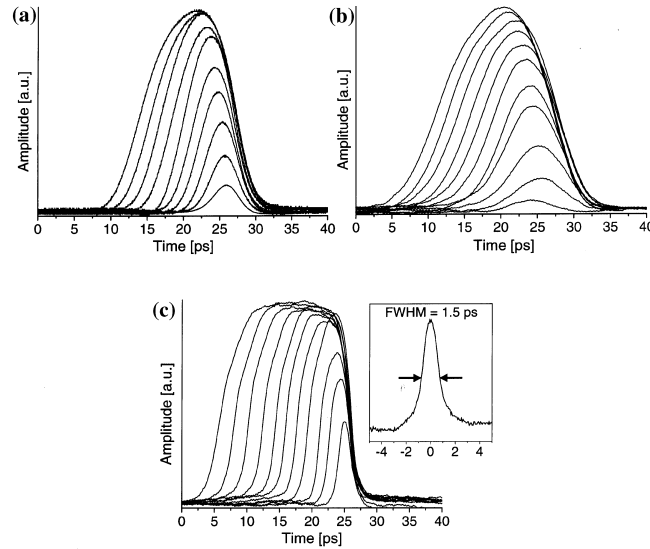


Fig. 3. Experimental measurements of the temporal switching window for the (a) TOAD, (b) CPMZ, and (c) SMZ as the window size is successively reduced. Inset: the shortest switching window achieved with the SMZ.

The SMZ architecture is best suited for high bandwidth applications. Due to its co-propagating nature, the switching window of the SMZ has the unique characteristic of both sharp rising and falling edges and therefore exhibits no dependence upon the length of the optical amplifiers. Ultrashort pulse widths of approximately 500 fs at a 2.5-GHz repetition rate were used to evaluate the device. The data and control pulse energies were set to 16 and 200 fJ respectively at the input ports of the switch. The results of the experiment are shown in Fig. 3(c). For  $\Delta t_{cs} > 6$  ps, the peak of the switching window flattens out, indicating that the data pulse is completely switched for the duration of the window. For smaller control pulse separations, the amplitude gradually decreases due to the finite temporal widths of both the control and data pulses. The smallest detectable window was 1.5 ps, as shown in the inset of Fig. 3(c). This performance indicates that the SMZ architecture may be suitable for demultiplexing from a 660-Gb/s data stream. Reducing the optical switching window further to subpicosecond regimes may be challenging due to SOA gain compression (Tajima *et al.* 2000). Further investigation and measurements of these dynamics are an important area of future research.

The efficiency of all of these optical switches can vary depending upon the device operating characteristics and the geometry used. For the switching windows described in this section, the flat portion of the window amplitude can correspond to a switching efficiency of greater than 100% due to the gain imparted on the data signal by the actively-biased optical amplifier. The SOA must be biased above the transparent point to achieve gain in the switched output. Experimental demonstrations have reported 6-dB gain in the switched output for a TOAD window width of 10 ps (Deng *et al.* 1998a). As the switching window is reduced, the contrast and efficiency both decrease due to the finite pulse widths and SOA lengths used in the device. For the TOAD, a 4 ps switching window used to demultiplex a single channel from a 250 Gb/s data stream achieved 14% efficiency at 1.3  $\mu\text{m}$  (Glesk *et al.* 1994). As the SOA fabrication and design techniques improve, greater switching efficiencies for shorter switching windows are expected.

## 2.4. INTEGRATED ALL-OPTICAL SWITCHES

While all of the geometries presented can be constructed from discrete components, practical, high performance all-optical switches for commercial systems will most likely take advantage of photonic integration technology. Various groups have integrated the Sagnac, Mach-Zehnder, and Michelson interferometer all-optical switch geometries. In order to fabricate these devices, both monolithic and hybrid technologies have been used. We briefly review the progress that has been made in this area.



The first monolithically integrated nonlinear Sagnac interferometer capable of demultiplexing from 20 to 10 Gb/s or 5 Gb/s was demonstrated by the Heinrich Hertz Institute (HHI) in 1996 (Jahn *et al.* 1996). Both the CPMZ and SMZ geometries have been integrated and subsequently demonstrated as high-speed demultiplexers by many groups (Jahn *et al.* 1995; Hess *et al.* 1998; Tajima *et al.* 2000; Wolfson *et al.* 2000). The Mach-Zehnder structures may be more practical to integrate than the Sagnac configuration since they do not require a large loop radius needed to avoid bending losses in the waveguides. The SMZ has the inherent advantage that it exhibits the shortest switching window in the co-propagating configuration. Finally, an SOA-based optical switch using an integrated Michelson interferometer was used to demonstrate demultiplexing from 20 to 5 Gb/s (Mikkelsen *et al.* 1995). The Michelson configuration may be a practical approach to integrated switching as anti-reflection coating is only applied to one side of the device and only two fiber-to-chip couplings are necessary for its operation as a demultiplexer (Mikkelsen *et al.* 1995).

The highest performance for integrated demultiplexers has been demonstrated using the Mach-Zehnder structures. Both the CPMZ fabricated by HHI and the SMZ fabricated by Alcatel have been used to demultiplex from 40 to 10 Gb/s (Jahn *et al.* 1995; Wolfson *et al.* 2000). The Alcatel monolithic SMZ is an all-active device, as all waveguides contain an active SOA element fabricated on the same substrate. This device improves the optical power requirements extending the input signal dynamic range by using current feedback control of the SOA pre-amplifiers (Wolfson *et al.* 2000). A high performance monolithically integrated SMZ was also demonstrated by a collaboration among the Swiss Federal Institute of Technology, University of Denmark, and France Telecom. This group achieved reliable demultiplexing from 80 to 10 Gb/s (Hess *et al.* 1998). The highest performance for an integrated SMZ to date has been achieved using a hybrid technique employed by NEC (Tajima *et al.* 2000). Fiber guides and passive silica waveguides are first fabricated onto a silicon planar lightwave circuit (PLC). The active SOA array chip is then flip-chip mounted onto the PLC. NEC used this chip to demonstrate demultiplexing from 168 to 10 Gb/s (Tajima *et al.* 2000). Further progress in photonic device integration will enable the deployment of high performance optical switches in commercial systems.

### 3. Ultrafast optical TDM networks enabled by all-optical demultiplexers

The first application of ultrafast optical switches follows historically from their use as optical demultiplexers. The first demonstrations of 100-Gb/s systems using TOADs constructed from discrete components are described in

this section. In addition to providing high bandwidth, these systems offer low latency and high scalability as compared to electronic packet switching fabrics.

### 3.1. PRACTICAL ALL-OPTICAL DEMULTIPLEXING DEMONSTRATIONS

The very short temporal switching windows opened by optically inducing nonlinearity in SOAs enable optical pulses to be individually selected from an aggregated frame of data pulses. Each switching event depletes the excited carriers in the SOA from the conduction band. In order for a subsequent switching event to take place, the carriers must be electrically pumped back into the excited state. This recovery time is approximately 100–500 ps. Since the recovery time generally determines the repetition rate of the switch, the TOAD is ideally applied to optical demultiplexing applications. In this regime, the switch is activated at the baseband rate of the receiving node while the aggregate data rate is many times faster than the processing electronics.

The first experiments to evaluate the performance of the TOAD consisted of aggregating several pulses in time to form an ultrafast optical TDM frame. Typically one pulse in the middle of the frame was modulated with a pseudorandom data pattern while the neighboring pulses were all set to one. The demultiplexer was used to select the modulated data signal from the ultrafast time frame. The first experiment demonstrated the TOAD's ability to switch pulses from a 50 Gb/s time frame to a baseband rate of 1.25 Gb/s (Sokoloff *et al.* 1993). This was subsequently followed by a 250 Gb/s demultiplexing experiment (Glesk *et al.* 1994) and error-free demultiplexing from a continuous 160-Gb/s data stream (Suzuki *et al.* 1994). Notable in all of these demonstrations was the low control pulse energy ( $< 1$  pJ) needed to perform the demultiplexing function.

With the successful development of optical demultiplexing, many new techniques have been used to enhance the performance of these devices. As the optical switching function is based upon gain saturation in an SOA, the repetition rate of the demultiplexing operation is somewhat limited by the recovery time of the amplifier. A novel optical biasing technique using CW light has significantly reduced this recovery time (Manning and Davies 1994). It has been estimated that this method may enable the optical switch to function at repetition rates approaching 100 GHz (Manning *et al.* 1994). Other demonstrations have shown that the TOAD can be successfully used to demultiplex many wavelengths simultaneously from an aggregated OTDM/WDM data stream (Mathason *et al.* 1999). In addition to these applications, new architectures are also emerging that may improve the performance. Dual wavelength operation has been demonstrated using a configuration known as a gain transparent (GT)-SOA switch (Diez *et al.* 1999a). This device uses a

data signal at a longer wavelength than the control so that the data is far from the band edge of the optical amplifier. The signal-to-noise ratio of the device is enhanced and the switching contrast at the output can be improved. The GT-SOA switch has been successfully applied as an add/drop multiplexer (Diez *et al.* 1999a) that can accommodate several wavelength channels from in an OTDM/WDM data stream (Diez *et al.* 1998).

### 3.2. A DISTRIBUTED PHOTONIC PACKET-SWITCHED NETWORK

Although electronic packet switching techniques have been able to meet the bandwidth requirements of network traffic so far, they are ultimately limited by the processing capabilities of electronic devices. In an electronic packet-switched network, optical technology is typically used only for purposes of transporting data from one node to the next, and the data undergoes optical-to-electronic and electronic-to-optical conversions at each node. Ultimately, one would like to completely avoid this conversion bottleneck by keeping the data in the optical domain throughout the entire routing process to reduce the packet routing latency (Toliver *et al.* 1998). This type of data communication, where packets are routed in an optically transparent manner, is referred to as *photonic packet switching*.

By exploiting optical TDM processing techniques, it is possible to construct an ultra-high capacity packet switch without resorting to complicated wavelength conversion technology that would be required for a photonic packet switch based on WDM technology. An example of a network based upon optical TDM techniques is the packet-switched optical networking demonstration, or POND network (Toliver *et al.* 1998), which is capable of processing 100-Gb/s packets. In order to minimize the complexity of the routing protocols and simplify the routing hardware, the POND testbed is based upon a regular distributed architecture, such as the 8-node ShuffleNet (Stone 1971) shown on the left side of Fig. 4.

An exploded view of the network node architecture is shown in Fig. 4. The network node consists of five primary subsystems: (1) an optical packet generation and compression subsystem to interface the node data rate to the much higher optical TDM network data rate (Deng *et al.* 1997a), (2) an all-optical header processor to extract routing information from the incoming packets, (3) an electronic routing controller to set the state of the routing switch, (4) an optical space switch to route the packets to the appropriate output port, and finally (5) a packet detection subsystem for decompressing received packets (Toliver *et al.* 1999). A short fiber optic processing delay buffer ( $< 10$  m) is required to store the optical packet while the electronic routing controller sets the state of the routing switch. Although electronic circuitry is used to make the routing decision, the packet remains in the

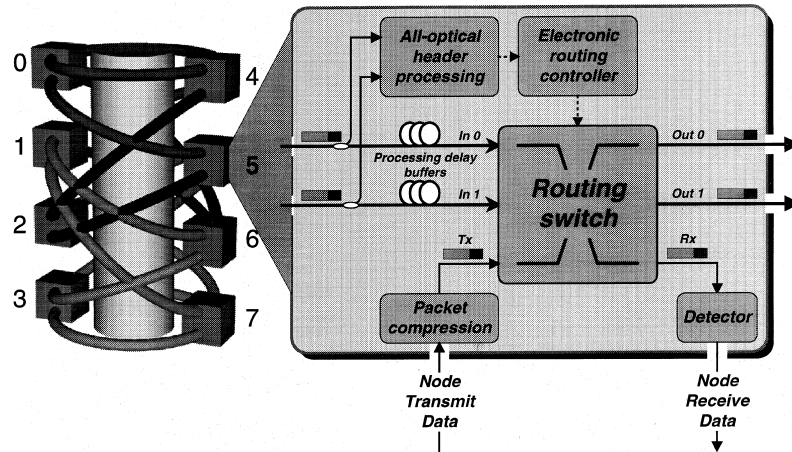


Fig. 4. An OTDM distributed photonic packet-switched network. An eight-node ShuffleNet architecture is depicted on the left. The exploded view of a node on the right shows the individual subsystems constructed for the demonstration.

optical domain throughout the entire routing process in this hybrid approach to photonic packet switching.

The photonic packet routing latency can be significantly reduced by keeping the number of address bits processed at each node to a minimum. In this case, the routing protocol developed for the POND network only requires two binary control bits in order to determine whether the packet should be directed to port Out 0, port Out 1, or received at the Rx port (Seo *et al.* 1997). The routing header that is appended to the front of a packet consists of a sequence of four such routing instructions, for a total of eight routing bits in the case of the 8-node ShuffleNet. Since this header is encoded at the same 100-Gb/s data rate as the packet payload, an array of two TOADs is used in the all-optical header processor to extract the routing control bits as the packet enters the node. For example, to route a packet from Node 0 to Node 7 in the POND network, a possible path is through intermediate Nodes 4 and 1. The packet output routing sequence for the path 0, 4, 1, 7 should be Out 0, Out 1, Out 1, and Rx respectively. The eight-bit header address sequence corresponding to this path is {00}, {10}, {10}, {11} (Toliver *et al.* 1998).

A single node of the POND network was constructed to serve as a testbed for evaluating the key optical technologies required to implement an entire network. To demonstrate the functional operation of the testbed, a sequence of 100-Gb/s packets is injected into the routing node to simulate the path through Nodes 0, 4, 1, and 7. Since only one node was constructed, the physical network node has a dynamic virtual address that is updated for each packet detected to reflect the simulated path through the network. The

routing control signals and switch outputs are monitored with photodetectors to verify that the optical packets are detected and routed correctly.

To test the entire routing path from Node 0 to Node 7, a sequence of four 100-Gb/s packets, separated by 160 ns, are injected into the node. The demultiplexed header bits are shown in Fig. 5(a). The top two waveforms, TOAD 0 and TOAD 1, are the TOAD output signals as measured on a bandwidth-limited photodetector. These outputs after digital thresholding are displayed in the two lower traces, Thr 0 and Thr 1. As each packet enters the node, a pair of header bits is demultiplexed in parallel by TOAD 0 and TOAD 1 from a specific routing group. In Fig. 5(a), the address sequence is correctly demultiplexed as {00}, {10}, {10}, and {11}. For example, both TOAD 0 and TOAD 1 demultiplex a '1' from the header of Packet 4 and read the address sequence {11}. These digital routing control bits are sent to the routing controller to set the state of a  $4 \times 4$  lithium niobate space switch. In this experimental demonstration, the routing latency was approximately 20 ns which was determined by the clock speed of the programmable EPLD used as the routing controller (Yu *et al.* 1998). By replacing this chip with a high performance application specific integrated circuit (ASIC), the routing controller decision time may be reduced significantly.

Photodetectors placed at the three outputs of the routing switch verify that the packets are routed correctly through the switch. The detected signals are shown in Fig. 5(b). Note that each 100-Gb/s packet appears as a single pulse due to bandwidth-limited detection. However, it can be seen from this figure that the packets are routed correctly according to the instructions in the header. Therefore, this experiment verifies the capability of performing photonic packet switching using control bits encoded in 100 Gb/s optical TDM packets.

To implement this optical TDM packet switching system on a larger scale, optical packet synchronization may be required to coordinate packet routing control. One technique for optically synchronizing the packet switching node uses a self-clocking strategy where an optical clock propagates with the high-

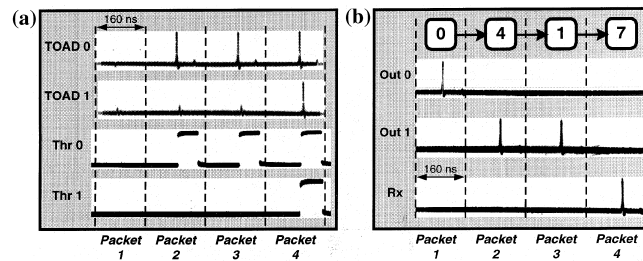


Fig. 5. Results of the packet switching demonstration: (a) the two-bit 100-Gb/s address decoded by the TOADs in the header processing array for four test packets injected into the network and (b) the output of the routing switch for the four 100-Gb/s test packets as displayed on a bandwidth-limited detector.

speed photonic packet during transport (Perrier and Prucnal 1989). The optical clock may be removed at the node upon arrival for bit-level synchronization by employing wavelength, amplitude, polarization, or time slot discrimination. By using a modified TOAD or other optical switch architecture, novel techniques have been developed to separate the clock from the packet based upon both amplitude (Deng *et al.* 1997b) and time slot (Toliver *et al.* 2000b). A self-clocked system achieves local synchronization despite the asynchronous arrival of high-speed optical TDM packets.

### 3.3. A BROADCAST-AND-SELECT ARCHITECTURE FOR ULTRAFast PHOTONIC SWITCHING

The promise of optics to relieve the bandwidth bottleneck in transmission systems was realized with the advent of DWDM for transport. However, as capacity along a single fiber strand has exceeded 3 Tb/s (Nielsen *et al.* 2000), a new bottleneck is emerging at the endpoints of this network where switching and routing is performed to direct packets to their destinations. Currently, the majority of backbone routers rely upon electronic crossbar switches to route packets. As the switching is performed in the electronic domain, optical packets are first received and buffered via electronics prior to switching. After the packets have been forwarded through the switching fabric, the packets are converted back into the optical domain for transport. The inherent speed limitations of the electronic-to-optical conversion as well as the high latency and low scalability of the electronic crossbar backplane limits the performance of the network and constrains the usable bandwidth in optical networks.

Recently, several experimental demonstrations (Barry *et al.* 1996; Tsukada *et al.* 1996; Lucek *et al.* 1997) have shown that optical TDM can meet many of the demanding needs of a high performance switching fabric which include full connectivity, low latency, high aggregate throughput, reliability, and scalability. By using our optical TDM approach, we constructed a testbed for a bit-interleaved 100-Gb/s switched interconnect based upon a broadcast star architecture (Deng *et al.* 1998a). Unique to our network is a highly scalable, novel node design that provides average inter-channel switching latency equal to the single channel bit period of 1.6 ns. This architecture is faster and more scalable than electronic crossbars and can be used as a switching fabric in the backplane of an enterprise switch or backbone router.

Fig. 6 shows the network and novel node architecture. The two key optical components of the node are the fast time slot tuner (Deng *et al.* 1997c) and the TOAD. A network interface card (NIC) sends electronic NRZ data at the single channel bit-rate,  $B$ , and control bits to a controller board specially designed to operate the two time slot tuners on the optical clock and data

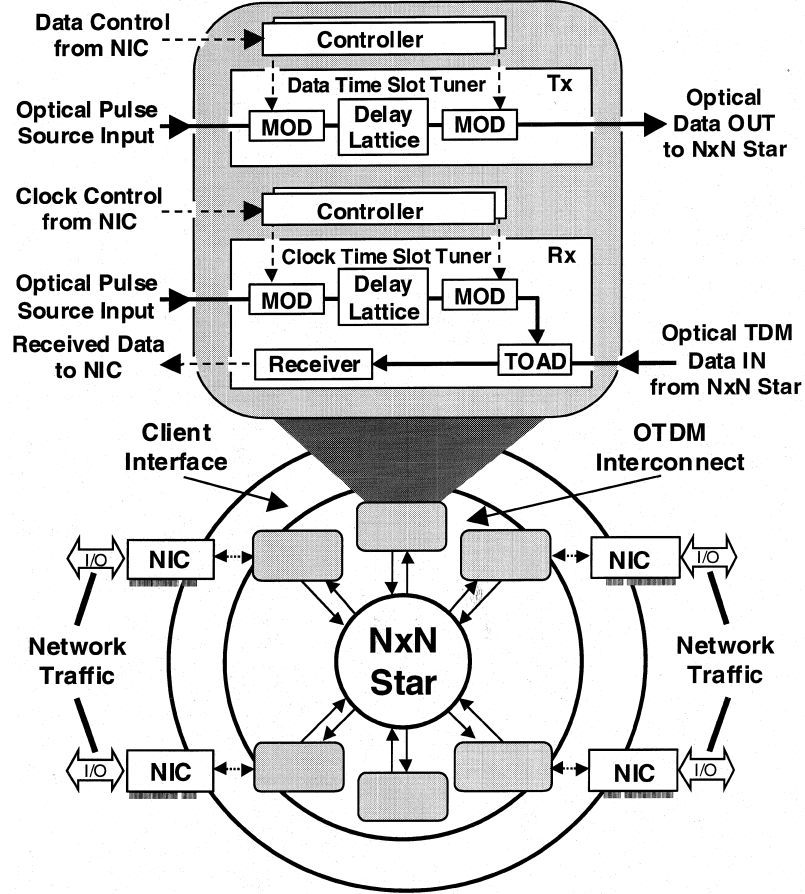


Fig. 6. A reconfigurable OTDM interconnect based upon a broadcast-and-select architecture. The individual architectural layers demonstrate its application as an optical router.

lines. The time slot tuners in the node are used in the transmitter to send modulated data into a given time slot within the optical TDM frame and in the receiver to align the optical clock with a given time slot for optical demultiplexing using the TOAD. To achieve global synchronization, picosecond pulses from a single fiber mode-locked laser source are amplified and distributed to the individual nodes via  $1 \times N$  splitters. After node data modulation and time slot selection, the data are multiplexed by precision fiber delays feeding an  $N \times N$  star coupler. The high bandwidth optical TDM frame is broadcast to all nodes in the network. The overall structure of the aggregated OTDM frame is determined by the time slot tuners used in the network nodes. For time slot tuners with  $k$ -stages of feed-forward delay lattices, the frame is arranged into  $2^k$  subcells of time duration  $T$  each containing  $2^k$  unique OTDM channels as shown in Fig. 7 (Deng *et al.* 1997c).

As a result, the network can support  $N = 2^{2k}$  nodes. The high-speed data rate of each subcell is  $\tau^{-1}$  which is determined by the precision time control of the delay lattices.

In our experimental testbed, we populated  $N = 16$  time slots in the optical TDM frame using  $k = 2$  stage time slot tuners. According to the design of the time slot tuner, the 1.6 ns frame is arranged into four subcells which each containing four 100-Gb/s time slots. The single channel data rate was chosen to match the OC-12 rate ( $B = 622.08$  Mb/s). The simple electronic design of the controller board permits the rapid control of the time slot tuners and provides low latency, arbitrary channel selection using a simple computer interface (Runser *et al.* 1999). Each TOAD was designed with a demultiplexing window width of about 10 ps at FWHM.

We constructed two fully functional nodes to measure the bit error rate (BER) and demonstrate the rapid inter-channel switching capability of the network nodes. These experiments were performed using adjacent channels in the same 100-Gb/s subcell. For average data and control input powers greater than  $-21$  dBm (13 fJ pulse energy) and  $-8$  dBm (250 fJ pulse energy) respectively, several hours of error free operation were achieved (Deng *et al.* 1998a). The fast inter-channel switching capability of the network was also demonstrated by using a previously proposed, low latency arbitration protocol (Nowatzky and Prucnal 1995) and two nodes of the network. The receivers of both nodes are fixed to listen to their own time slots. Each node first transmits its binary address at the single OC-12 channel rate into its own time slot. If successfully received, each node then transmits its address into the time slot of the other node by reconfiguring the data time slot tuner. Fig. 8 shows a demonstration of the protocol using two nodes in the network whose time slots are adjacent in the 100-Gb/s subcell. The addresses assigned to Node 0 and Node 1 were 0101 and 0111 respectively. The traces shown are the demultiplexed TOAD outputs directly from the output of the receivers

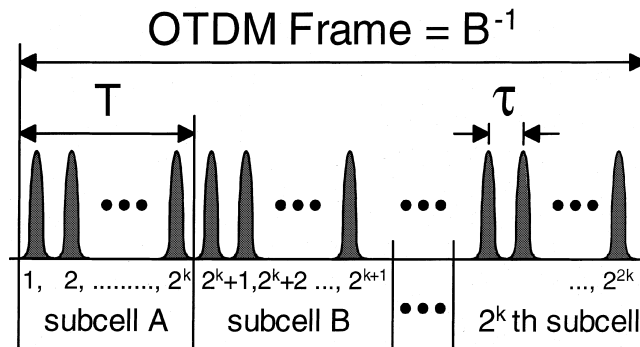


Fig. 7. Time slot allocation in an OTDM interconnect based upon a time slot tuner with  $k$ -stages of fiber delay lattices.



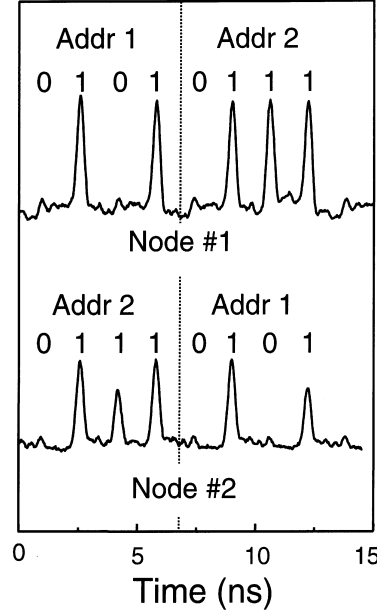


Fig. 8. Demonstration of rapid time slot switching between two nodes using the OTDM interconnect in a fixed receiver configuration. The average hardware latency of the operation was 1.6 ns.

for the two nodes. After each node successfully receives its own address, the data time slot tuners rapidly reconfigure within a single bit period to transmit into the time slot of the other node. Note that each node now successfully receives the address of the other in its own time slot. The average time slot switching hardware latency using this technique and the controller board was 1.6 ns.

This architecture represents a practical and scalable alternative to electronically switched backplanes. In addition to single-channel access, we have demonstrated advanced services such as multicasting using the same optical subsystems and components (Deng *et al.* 2000). Since the active components in the time slot tuners do not scale with the number of nodes (Deng *et al.* 1997c), simply adding another stage,  $k = 3$ , scales the interconnect up to  $N = 64$  nodes without taxing the power budget significantly. If OC-24 ( $B = 1.24416$  GHz) is chosen as the single channel data rate and 10-GHz intermediate processing electronics are used, an 80-Gb/s interconnect with an average rapid inter-channel switching speed of 800 ps is feasible. In such a 64-node architecture, coherent crosstalk does not limit the BER performance significantly (Deng *et al.* 1998b). Ultimately, the dimensionality of the network is determined by the coherent crosstalk. Since the demultiplexer and other optical components in the node can be integrated, we believe this network is practical for future, high-speed multiprocessor interconnect systems and high performance switching fabrics.

#### 4. All-optical packet routing using the TOAD

To alleviate the electronic bottleneck imposed upon present-day routing switches, all-optical switching techniques provide a novel solution. We discuss a  $1 \times 2$  self-routing optical packet switch that can be cascaded to form more complex, transparent optical switching fabrics with low latency.

##### 4.1. $1 \times 2$ SELF-ROUTING ALL-OPTICAL PACKET SWITCH

Although the previous architectures largely address the rapid switching capability of optical TDM systems, the interconnects still rely upon some level of electronic control. In the case of the optical packet switch described in Section 3.2, electronic signals are required to trigger LiNbO<sub>3</sub> optical switches in the crossbar. The broadcast-and-select architecture in Section 3.3 also uses electronic control of the tunable delay elements to achieve time slot tuning. Maximum throughput with minimum delay can be achieved in transparent all-optical networks where optical address recognition is used to trigger an all-optical packet routing switch. This approach alleviates all dependence upon electronic processing and achieves the minimum latency.

By combining all-optical packet header recognition with an all-optical packet switch, a transparent, self-routing optical node can be constructed. The schematic design of a  $1 \times 2$  self-routing optical node is shown in Fig. 9. Optical TDM packets enter the input port where a polarization splitter (PS) is used to separate the optical clock from the data. The first TOAD (TOAD1) is set with a short window ( $< 10$  ps) to demultiplex a single address bit from the incoming packet header using the extracted packet clock pulse. This address bit is used to control a second TOAD (TOAD2) which is configured as a routing switch. By simply adjusting the offset position of the SOA from the center of the loop, the size of the switching window can be increased to switch the multiple pulses contained in the packet to the output port. A '1' bit detected in the packet address at TOAD1 causes the packet to be optically switched to the output port (OUT 1) of the packet routing switch. When the address bit is '0', the packet is reflected at TOAD2 and coupled to a secondary output (OUT 2). This method allows an all-optical node to transparently route packets to one of two output ports simply by reading a single bit from the address. Synchronization of the switch is achieved by using a self-clocking scheme where the optical clock propagates with the packet in an orthogonal polarization. To account for the all-optical address processing delay, a short fiber length is used to buffer the packet prior to entering the input port of TOAD2. An SOA is used at the output of TOAD1 to amplify the demultiplexed address pulse to a sufficient level to trigger the switching operation of TOAD2. An optical isolator (OI) is inserted between TOAD1

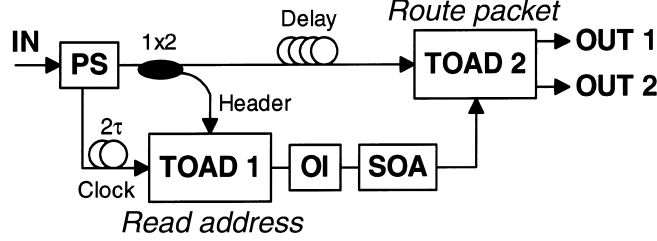


Fig. 9. Schematic of a  $1 \times 2$  all-optical packet switch using two cascaded TOADs for address recognition and packet routing.

and the inline SOA to reduce optical feedback within the system. Cascading multiple  $1 \times 2$  optical nodes enables the construction of large scale, Banyan-type transparent routing networks.

#### 4.2. EXPERIMENTAL DEMONSTRATION

To demonstrate one bit all-optical address recognition and packet switching using the self-routing scheme, the node described in Fig. 9 was constructed (Glesk *et al.* 1997). The offset position for the SOA in TOAD1, which performs the address recognition, is described by  $\Delta x_{\text{TOAD1}} = \tau \cdot c_{\text{fiber}}/2$  where  $1/\tau$  is the bit-rate of the incoming data. The switching window for TOAD2 is much wider so that it can accommodate an entire optical packet. The offset position for the SOA in TOAD2 is described by  $\Delta x_{\text{TOAD2}} = T \cdot c_{\text{fiber}}/2$  where  $T = N\tau$  and  $N$  is the number of bits in the packet.

For the experimental demonstration,  $1/\tau = 250\text{-Gb/s}$  packets were generated from 1-ps pulses emitted from a Nd:YLF laser followed by a compression stage. Fig. 10 shows the timing diagram and experimental detection of the four-bit test packets used in the experiment. The first bit is a clock pulse orthogonally polarized to the subsequent data pulses in the packet. A fixed delay after the polarization splitter at each node is used to align the clock with the appropriate address bit in the header. In this fashion, a packet will flow through the network and each node will read a unique bit from the header to route the packet to its destination. In this experiment, a delay of  $2\tau$  is used so that address bit 2 routes the packet through the node. This bit was modulated with a '1' or a '0' to test the operation of the  $1 \times 2$  all-optical routing switch. Since these bits are spaced at an interval of 4 ps, a bandwidth-limited detector cannot distinguish the individual bits of the header. As a result, the two input packets appear as 'triple height' and 'double height' in Fig. 10 for header patterns '111' and '101' respectively. The timing diagram and results of the routing experiment are shown in Fig. 11. Packets with a '1' at header bit position 2 are switched by TOAD2 to OUT 1 whereas packets

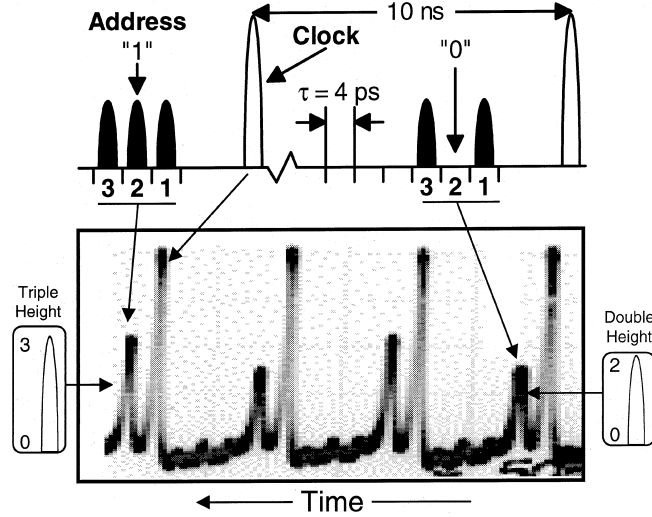


Fig. 10. Timing diagram and experimental measurement of the packets used to demonstrate the  $1 \times 2$  optical packet switch. The individual pulses in the 250 Gb/s packets are not resolved on the bandwidth-limited detector but the packets can be distinguished by their aggregate amplitude.

with a '0' in bit position 2 are reflected by TOAD2 and coupled to OUT 2. Note that the 'triple height' packets are correctly switched to OUT 1, and the 'double height' packets are reflected to OUT 2.

#### 4.3. DISCUSSION AND FUTURE WORK

The two TOADs in the cascaded configuration provide the key functionality for the optical self-routing switch. While the first TOAD uses a switching

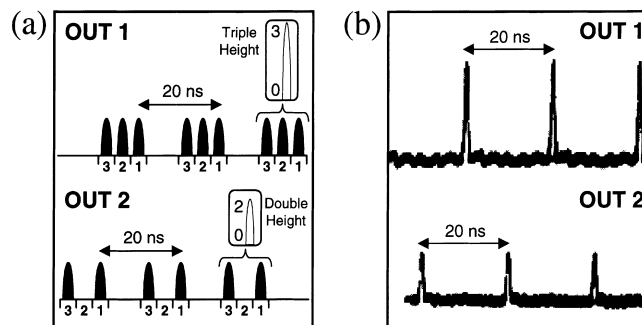


Fig. 11. Results of the optical packet routing experiment using the  $1 \times 2$  packet switch. (a) timing diagram of the received output and (b) output of the  $1 \times 2$  switch as seen on a bandwidth limited detector.

window that is much shorter in duration than the SOA recovery time, the TOAD performing the packet switching operation has a window duration which may approach or exceed the recovery time. Packets that contain many bits may represent a challenge to the system. Consider a 1000-bit packet at a bit-rate of 250 Gb/s. The packet time of 4 ns is approximately 10 times the typical recovery time of the SOA. As the SOA recovers, the amplitude of the switching window decays exponentially (Toliver 2000a). This may adversely affect the individual bits of the packet payload and cause an amplitude reduction from the beginning to the end of the packet.

To accommodate longer packets, other techniques may be applied to maintain a flat, uniform switching window for the duration of the packet. One scheme involves clocking TOAD2 with several control pulses for each packet. Clocking the optical switch every 100 ps, for instance, maintains a relatively flat and open switching window for the duration of the packet. Another approach is to modulate the electrical bias of the SOA. The current pumping the carriers in the SOA can be decreased just after the switching event has been initiated to prevent a rapid recovery. This technique permits the switching window to remain open for a longer period of time with more uniform amplitude.

Future, compact  $1 \times 2$  routing switches may be integrated onto the same substrate using the integration techniques previously described in Section 2.4. Large-scale integration interconnecting many of these switches enables the construction of self-routing optical switches with many ports.

## 5. Data format and wavelength conversion

The optimal data format for transmission links or switching systems depends on parameters such as the overall dispersion, launch power, wavelength spacing, and bit rate (Sano *et al.* 1998). Depending on the specific application, the optimal data format may be either non-return-to-zero (NRZ) or return-to-zero (RZ) with a certain pulse width or duty cycle. An optical pulse width management device (OPWMD) can be a powerful tool to achieve the desired data format through all-optical format conversion.

Format conversion from NRZ to RZ and RZ to NRZ in fiber optic communication systems can be accomplished all-optically using the TOAD. By changing the configuration of the data and control input ports and varying the size of the switching window, the desired format conversion can be achieved. In addition to its use as a pulse width management device, the TOAD can also function as a wavelength converter if the wavelength at the data input port differs from that at the control port. The progress made using the TOAD to perform pulse compression (NRZ to RZ), stretching (RZ to NRZ), and wavelength conversion will be reviewed this section.

### 5.1. PULSE COMPRESSION, NRZ TO RZ CONVERSION

To perform NRZ to RZ conversion, the TOAD is used to sample an NRZ modulated data stream. The incoming NRZ data stream enters the data input port of the TOAD while the control input is a picosecond optical clock. The TOAD effectively samples the NRZ data stream which ‘compresses’ the data to the width of the switching window. The duty cycle of the new RZ data stream can be controlled by changing the size of the switching window.

To experimentally demonstrate data pulse compression, an NRZ data stream at the TOAD input port is generated by inserting a LiNbO<sub>3</sub> electro-optic modulator between a CW laser source and the TOAD data input port. Picosecond pulses are injected into the control port without any modulation. Each control pulse saturates the SOA and opens a switching window. If a ‘0’ bit is present in the NRZ data stream, nothing is switched to the output port. However, a portion of the ‘1’ bit in the NRZ input data stream will be switched to the TOAD output port. The size of the pulse that is switched to the output is determined by the size of the switching window. Fig. 12(a) shows the results of the NRZ to RZ format conversion demonstration. The upper trace shows the NRZ signal and the lower trace shows the corresponding RZ signal after conversion. The shortest RZ data pulse obtained was 4 ps. Bit-error-rate (BER) measurements for pulse compression from NRZ to RZ are shown in Fig. 13 for an OC-48 NRZ  $2^{31} - 1$  pseudorandom data sequence. The results show a slight power penalty for the shorter switching window of 13.7 ps. This behavior is expected as the reduction of the optical switch output amplitude as the switching window is reduced in temporal width causes fewer photons to be switched to the output.

### 5.2. PULSE STRETCHING, RZ TO NRZ FORMAT CONVERSION

Conversion from RZ to NRZ can be performed through a different configuration. An RZ modulated data stream enters the *control* input port of the

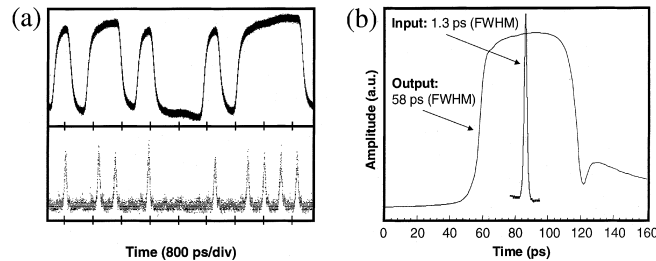


Fig. 12. All optical format conversion results: (a) conversion of an NRZ OC-48 16-bit data pattern (top) to RZ data format with 4 ps pulses (bottom) as shown on a bandwidth-limited detection system, and (b) RZ to NRZ conversion or pulse stretching of a 1.3 ps pulse (as measured by autocorrelation) to 58 ps.

TOAD while the signal at the data input port is a CW beam. If the switching window duration is set to the data bit period, the output of the TOAD will be an NRZ data stream with the corresponding data modulation.

To experimentally verify pulse stretching, a tunable delay element is inserted into the TOAD loop to adjust the displacement of the SOA from the center of the loop. A CW diode laser is connected to the input data port. The incoming RZ data stream is obtained by modulating the output of a fiber modelocked laser, which generates 1.3-ps pulses. This RZ data stream is injected into the TOAD control port. When the CW light enters the data input port of the TOAD, the light is split into two halves that travel in clockwise and counter-clockwise directions. When there is a '0' in the RZ stream at the control port, the TOAD acts as a loop mirror. The CW light traveling in both directions experiences the same gain and phase and is reflected back toward the source. However, when a '1' is present in the RZ data stream, the pulse saturates the SOA and creates a switching window, which causes the CW light in the window to be switched to the output port.

The RZ picosecond pulse stream can be stretched into a much broader NRZ stream by setting the SOA offset appropriately. For instance, the results of stretching pulses from 1.3 to 58 ps as shown in Fig. 12(b) indicate the ability to perform RZ to NRZ conversion on data streams approaching 20 Gb/s. The 1.3-ps pulse in the Fig. 12(b) is obtained by measuring the autocorrelation of the input control pulses and the 58-ps pulses were obtained by measuring the switching window after setting the tunable delay element appropriately. To demonstrate the reliability of the device, two

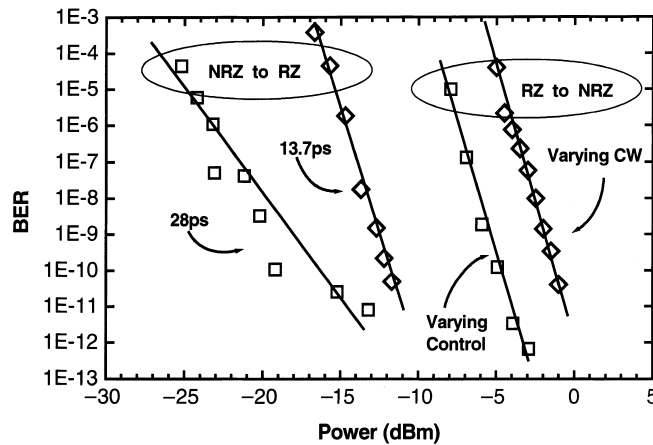


Fig. 13. Bit error rate measurements for both NRZ to RZ (pulse compression) and RZ to NRZ (pulse stretching) format conversion. For NRZ to RZ, two switching window sizes of 28 ps (squares) and 13.7 ps (diamonds) were investigated. For RZ to NRZ, the BER was measured by fixing the CW input power and varying the control power (squares) and also by fixing the control power and varying the CW input power (diamonds). Simultaneous wavelength conversion was also achieved with the RZ to NRZ format conversion.

different BER measurements were performed for RZ to NRZ conversion as shown in Fig. 13. For a fixed and optimal CW input power, the control power was varied to produce the BER shown with the square symbols. For the other measurement indicated with diamond symbols, a fixed control power was used while the CW input power varied. The curves demonstrate that a BER of  $< 10^{-11}$  can be achieved for either operating regime.

### 5.3. WAVELENGTH CONVERSION

Wavelength conversion can be accomplished using the TOAD if the wavelength of the optical input signal differs from that of the control signal. Format conversion from RZ to NRZ can be achieved with simultaneous wavelength conversion, for instance, if an RZ data stream entering the control port of the TOAD has a different wavelength than that of the CW source at the data input port. In this case, the new NRZ modulated data stream will be imprinted onto the wavelength of the CW source. The converted wavelength at the output port also maintains the same data polarity as the original RZ data stream. This wavelength conversion technique has less pattern dependent effects and is capable of operating at a high bit rate (Cotter *et al.* 1999). In addition, the signal-to-noise ratio of the data on the new wavelength is significantly enhanced since the data on the noisy input stream is imprinted on a clean local optical source (Zhou 2000). We obtained reliable operation of simultaneous wavelength and RZ to NRZ conversion with a BER of  $10^{-11}$  as shown in Fig. 13.

## 6. High bandwidth optical sampling systems

As communications technology moves toward higher bit-rate systems, communications test and measurement equipment will require bandwidths exceeding 100 GHz. While such bandwidths cannot be practically achieved using present day electronic measurement systems, new optical techniques may offer a solution. In addition to diagnostic measurements of high-speed optical communication signals, measurements of short optical waveforms are useful in laser spectroscopy, fiber optic sensors, optical ranging, and biomedical optical tomography. These measurements require high temporal resolution and real-time monitoring of the received optical waveform. Many techniques have been used to measure ultrashort optical waveforms. Streak cameras are used extensively in research but are much too bulky and costly for commercial systems. Traditional optical correlation techniques that rely upon optical nonlinearity in bulk materials require high optical powers. These systems typically require a repetitive optical signal and are not suitable



for single-shot, arbitrary waveforms. Nonlinear loop mirrors have also been used extensively as optical samplers but require long fiber lengths and high control pulse powers for sufficient detection of the sampled signal (Nelson and Doran 1991). New approaches to optical sampling using nonlinearities in SOAs have recently been developed by several groups to address these application areas and alleviate some of the limitations to traditional optical techniques (Jinno *et al.* 1994; Kawaguchi and Inoue 1998; Diez *et al.* 1999b).

As we have previously demonstrated, the TOAD is an ideal candidate for commercial optical systems due to its low control pulse energy and integration potential. This section discusses two system demonstrations that apply the TOAD as an analog optical sampling gate. In each system, novel devices were used to introduce time dilation between the sampling window and the signal waveform. Both single-shot and real-time operation of the optical sampling methods are presented.

#### 6.1. A SINGLE-SHOT OPTICAL SAMPLING SYSTEM

Single-shot optical measurement systems are especially important for detecting and analyzing non-repetitive, analog optical waveforms. These signals may be generated from a bursty optical source directly or represent a high bandwidth analog electrical waveform modulated onto an optical carrier. We have developed a sampling system capable of detecting single-shot optical waveforms with intensity envelope bandwidths exceeding 100 GHz (Deng *et al.* 1998c). In addition to employing low bandwidth detection electronics, the system uses an optical control signal with  $< 1$  pJ of pulse energy. The two main components of the system shown in Fig. 14 are the retarding pulse replicator and a TOAD used as an ultrafast sampling gate. The incoming single-shot optical signal is first replicated into successive copies according to the repetition rate of the local sampling clock. Each signal copy is then fed into the input of a TOAD with a short sampling window. The interval between the signal copies is precisely delayed in time with respect to the sampling clock such that each sampling operation takes place at a different portion of the signal. Because the waveform is sampled at the clocking rate of the TOAD, only low-bandwidth devices are required to display the sampled output.

Through copying and delaying the incoming signal, the retarding pulse replicator provides time dilation between the sampling clock and analog waveform. The replicator consists of cascaded stages of fiber lattices (Deng *et al.* 1997a). In the experimental demonstration a  $k = 4$  stage replicator structure was chosen to make  $2^k = 16$  copies of the incoming waveform. The output of a 100-MHz Nd:YLF laser at 1313 nm with a 31-ps pulse width (FWHM) was modulated to produce a slower, 'single-shot' signal pulse rate

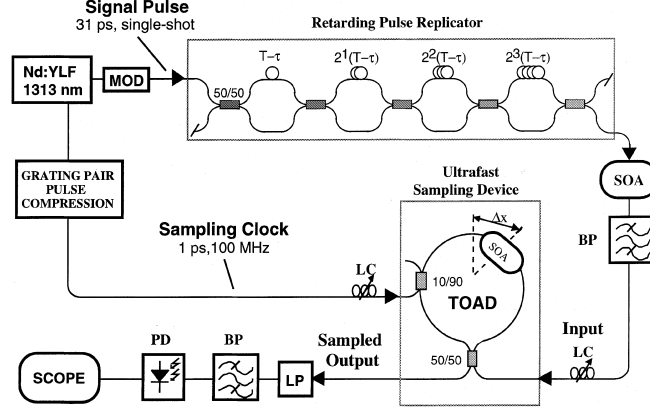


Fig. 14. Schematic of a single-shot all-optical sampling oscilloscope.

of 3.125 MHz. By splitting this pulse and passing it through a grating-pair compression stage, a 1-ps optical clock synchronized with the signal waveform was also generated. The raw sampling rate was fixed at the laser repetition frequency of 100 MHz for the sampling clock giving  $T = 10$  ns. The sampling window was chosen to be 7 ps at FWHM and the temporal offset,  $\tau$ , per replicator stage was set to 10 ps. The output of the pulse replicator was amplified by an SOA to compensate for the losses through the structure. A bandpass filter following the inline SOA blocked broadband spontaneous emission. Finally, the TOAD was electrically biased at 45 mA and optically clocked with 1-ps pulses with an energy of 600 fJ. Polarization discrimination was used at the output of the TOAD to distinguish the sampled signal from the clock.

To demonstrate the single-shot capability of the system, the 31-ps signal pulse was split unevenly by a fiber coupler, and recombined such that the two pulses were separated by 80 ps to create an arbitrary waveform. Fig. 15(a) shows the double pulse waveform directly detected by a 20-GHz photodetector followed by a 50-GHz oscilloscope. From this trace, it is difficult to distinguish the two pulses from each other. The pulses, however, are successfully resolved by the sampling system as shown in Fig. 15(b). In contrast to the direct detection method, the bandwidth of the photodetector and oscilloscope following the TOAD output was less than 150 MHz.

By choosing  $T$ ,  $\tau$ , and  $k$  appropriately, this system provides a scalable architecture for a variety of sampling applications. For this demonstration, a system capable of sampling waveforms up to  $2^k = 160$  ps in duration with a bandwidth greater than 100 GHz is possible. Since  $T = 10$  ns, the next single-shot event can be sampled and monitored in real time after a  $2^k T = 160$  ns delay. In future designs, the raw sampling rate and temporal resolution of the system can be improved. Using a laser with a higher repetition rate, the raw

## INTERFEROMETRIC ULTRAFAST SOA-BASED OPTICAL SWITCHES

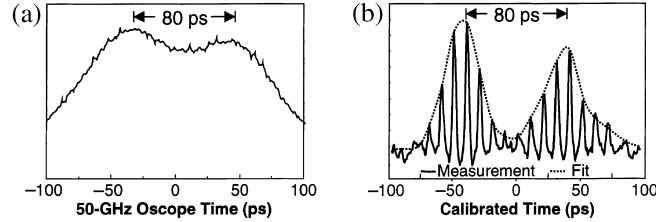


Fig. 15. Results of the single-shot optical sampling oscilloscope: (a) two pulses as seen on a 20-GHz bandwidth-limited detector and (b) the pulses successfully resolved using the single-shot sampler.

sampling rate may be increased to more than 10 GHz, limited only by the SOA recovery time. Likewise, the time delay between real-time sampling events of consecutive waveforms is reduced. We expect that a future system employing an integrated device will be capable of sampling optical waveforms with bandwidths approaching 1 THz.

### 6.2. A CONTINUOUS, REAL-TIME OPTICAL SAMPLING SYSTEM

While the previous system is an excellent candidate for single-shot waveforms, detection of repetitive signals such as those in a communication system are also an important application for optical test and measurement. The goal of real-time optical sampling is to detect continuous, repetitive signals using time dilation techniques similar to those in electronic oscilloscopes. This system provides further flexibility and is more practical for developing low cost, high performance sampling systems.

The real-time, ultrafast optical sampling system is shown in Fig. 16. The system uses a free-space mechanical vibrator in the path of the optical clock to generate a continuous time delay between the optical signal and the clock triggering the sampling operation in the TOAD. The mechanical vibrator simply consists of a corner-cube mirror attached to the center of an 8-inch speaker and oscillates at a frequency of 10 Hz. By continuously varying the optical clock over this range, a portion of the optical signal can be extracted and displayed on a low bandwidth oscilloscope triggered at the vibrator oscillation frequency.

To experimentally demonstrate the system (Runser *et al.* 2000), the output of a modelocked erbium-doped fiber laser (ML-EDFL) was split and amplified to generate synchronized optical clock and signal pulses. The ML-EDFL produces 2.8-ps pulses with a hyperbolic secant profile at 1.55  $\mu\text{m}$  with a repetition rate of 10 GHz. The signal path is passively upconverted from a baseband repetition rate of 10–160 GHz using a four-stage feed-forward fiber delay lattice and introduced into the signal input of the optical sampling gate. The energy of each signal pulse in the train was only 1.5 fJ.

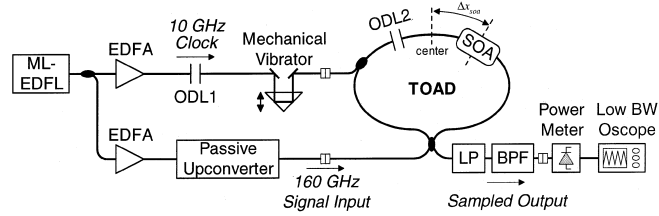


Fig. 16. Schematic of a real-time optical sampling system for repetitive optical signal.

The 10-GHz optical clock that triggers the sampling operation first passes through the mechanical vibrator, which varies the optical delay over a 40-ps range. With the SOA in the TOAD biased at 130 mA, a clock pulse energy of only 20 fJ is required to trigger the sampling operation. The TOAD sampling window was set to a width of 4.7 ps at FWHM. An optical average power meter with an update rate of 700 Hz receives the optically sampled signal at the TOAD output. An oscilloscope with a 100-MHz bandwidth is used to display the power meter output in time.

Through this sampling technique, the oscilloscope effectively displays the time dilation of the convolution of the TOAD transfer function with the high repetition rate optical input signal. The experimental results obtained using this sampling technique with the continuous 160-GHz pulse train are shown in Fig. 17. As the scanning range of the mechanical vibrator is only 40 ps, a long-range optical delay line (ODL1) was used to coarsely adjust the timing of the optical clock to view other portions of the waveform. Snapshots from the oscilloscope were concatenated to form the trace in Fig. 17. The sampling system effectively distinguishes the individual pulses in the 160-GHz pulse train. Although the peak amplitude of each pulse in the train is relatively the same, the minimum level is somewhat non-uniform. This is attributed to small timing errors in the fiber lattice multiplexer that cause deviations in the pulse positions. The output of the sampling system was compared to a slow, long-range statistical cross-correlation measurement. The error in the 160-GHz pulse positions from the expected 6.25-ps temporal spacing was determined using the two techniques. The standard deviation in pulse position measured using each method was 0.3 ps, which is on the order of the laser timing jitter.

This sampling system enables real-time evaluation of repetitive, high bandwidth, low energy optical signals using a compact all-optical sampling gate and simple, low bandwidth electronics. Since only 20 fJ of optical control energy is needed to trigger the sampling operation, we believe this system offers a practical alternative to traditional correlation techniques that rely upon bulk optical nonlinearities. By choosing the rate of the mechanical vibrator appropriately, real-time signals can be displayed in the presence of

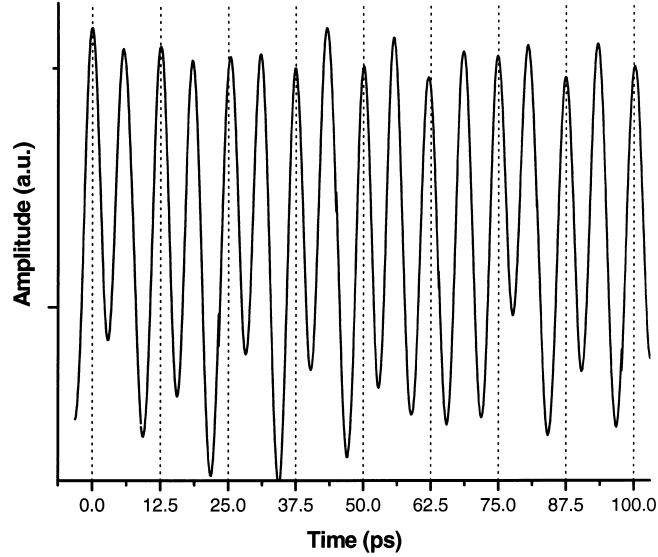


Fig. 17. Measurement of a 160-GHz optical pulse train using the real-time optical sampling technique.

experimental instabilities such as fluctuating free space fiber-to-chip couplings, thermal variations within interferometers, and other environmental irregularities. Faster scan rates and longer scan times can be accommodated by replacing the mechanical vibrator with a rotating linear delay element capable of rapidly sweeping the clock signal over a delay path of more than 100 ps (Optical Delay Generator FR-203, Femtochrome Research, Berkeley, CA). Higher sampling bandwidths approaching 1 THz are possible by using other SOA-based optical switch architectures that have previously been discussed. Since the TOAD sampling window is asymmetric in the time domain, a linear sampling system may employ signal processing and digital filtering to account for the non-ideal sampling function. However, the Mach-Zehnder geometries previously discussed have symmetric sampling functions that may be easier to implement into a sampling system. The effect of the optical switch sampling functions for all of the optical switch geometries is the subject of current investigation.

## 7. Conclusions

We have presented several applications of all-optical switches based upon SOA nonlinearities. Due to the low optical energy requirements, we believe these devices represent a practical approach to future, high-speed systems

employing all-optical signal processing. Successful demonstrations of optical demultiplexing, packet routing, simultaneous wavelength and format conversion, and optical sampling are some of the functions that these devices can perform using the same device geometry. High-speed optical networks and communication measurement systems are just a few of the areas where these devices may have a significant impact. Although some optical switch architectures have already been integrated, the future commercial potential of these devices depends upon further progress toward cost-effective photonic integration and packaging techniques. Pushing the optical switching bandwidth to 1 THz and beyond continues to be a challenging area for future research. Further exploration of semiconductor optical nonlinearities with ultrafast response times ( $< 1$  ps) such as electron spin polarization and intersubband transitions (Wada 2000) may enhance the switching speed of these interferometric devices or yield an entirely new class of all-optical switches capable of meeting the demand for bandwidth in future optical TDM systems.

## References

- Asobe, M., T. Oohara, I. Yokohama and T. Kaino. *Electron. Lett.* **32** 1396, 1996.
- Barry, R.A., V.W.S. Chan, K.L. Hall et al. *IEEE J. Sel. Areas Commun.* **14** 999, 1996.
- Cotter, D., R.J. Manning, K.J. Blow, A.D. Ellis, A.E. Kelly, D. Nesses, J.D. Phillips, A.J. Poustie and D.C. Rogers. *Science* **286** 1523, 1999.
- Deng, K.-L., K.I. Kang, I. Glesk, P.R. Prucnal and S. Shin. *Electron. Lett.* **33** 1237, 1997a.
- Deng, K.-L., I. Glesk, K.I. Kang and P.R. Prucnal. *IEEE Photon. Technol. Lett.* **9** 830, 1997b.
- Deng, K.-L., I. Glesk, K.I. Kang and P.R. Prucnal. *IEEE Photon. Technol. Lett.* **9** 1496, 1997c.
- Deng, K.-L., R.J. Runser, P. Toliver, C. Coldwell, D. Zhou, I. Glesk and P.R. Prucnal. *Electron. Lett.* **34** 2418, 1998a.
- Deng, K.-L., I. Glesk, K.I. Kang and P.R. Prucnal. *IEEE Photon. Technol. Lett.* **10** 1039, 1998b.
- Deng, K.-L., R.J. Runser, I. Glesk and P.R. Prucnal. *IEEE Photon. Technol. Lett.* **10** 397, 1998c.
- Deng, K.-L., R.J. Runser, I. Glesk and P.R. Prucnal. *IEEE Photon. Technol. Lett.* **12** 558, 2000.
- Diez, S., R. Ludwig and H.G. Weber. *Electron. Lett.* **34** 803, 1998.
- Diez, S., R. Ludwig and H.G. Weber. *IEEE Photon. Technol. Lett.* **11** 60, 1999a.
- Diez, S., R. Ludwig, C. Schmidt, U. Feiste and H.G. Weber. *IEEE Photon. Technol. Lett.* **11** 1402, 1999b.
- Doran N.J. and D. Wood. *Opt. Lett.* **13** 56, 1988.
- Doran, N.J., D.S. Forrester and B.K. Nayar. *Electron. Lett.* **25** 267, 1989.
- Eiselt, M. *Electron. Lett.* **28** 1505, 1992.
- Eiselt, M., W. Pieper and H.G. Weber. *Electron. Lett.* **29** 1167, 1993.
- Ellis, A.D., D.M. Patrick, D. Flannery, R.J. Manning, D.A.O. Davies and D.M. Spirit. *J. Lightwav. Technol.* **13** 761, 1995.
- Glesk, I., J.P. Sokoloff and P.R. Prucnal. *Electron. Lett.* **30** 339, 1994.
- Glesk, I., J. Sokoloff and P.R. Prucnal. *Electron. Lett.* **30** 1322, 1994.
- Glesk, I., K.I. Kang and P.R. Prucnal. *Electron. Lett.* **33** 794, 1997.
- Hall, K.L., G. Lenz, A.M. Darwish and E.P. Ippen. *Opt. Commun.* **111** 589, 1994.
- Hess, R., M. Caraccia-Gross, W. Vogt, E. Gamper, P.A. Besse, M. Duelk, E. Gini, H. Melchoir, B. Mikkelsen, M. Vaa, K.S. Jepsen, K.E. Stubkjaer and S. Bouchoule. *IEEE Photon. Technol. Lett.* **10** 165, 1998.
- Hong, M.Y., Y.H. Chang, A. Dienes, J.P. Heritage and P.J. Delfyett. *IEEE J. Quantum Electron.* **30** 1122, 1994.

# INTERFEROMETRIC ULTRAFAST SOA-BASED OPTICAL SWITCHES

- Jahn, E., N. Agrawal, M. Arbert, H.-J. Ehrke and D. Franke. *Electron. Lett.* **31** 1857, 1995.
- Jahn, E., N. Agrawal, W. Pieper, H.-J. Ehrke, D. Franke, W. Furst and C.M. Weinert. *Electron. Lett.* **32** 782, 1996.
- Jinno, M. and T. Matsumoto. *Opt. Lett.* **16** 220, 1991.
- Jinno, M., J.B. Shlager and D.L. Franzen. *Electron. Lett.* **30** 1489, 1994.
- Kawaguchi, H. and J. Inoue. *Proc. of SPIE* **3283** 477, 1998.
- Kang, K.I., T.G. Chang, I. Glesk and P.R. Prucnal. *Appl. Opt.* **35** 417, 1996a.
- Kang, K.I., I. Glesk and P.R. Prucnal. *Intl. J. High Speed Electron. Sys.* **7** 125, 1996b.
- Kang, K.I., T.G. Chang, I. Glesk and P.R. Prucnal. *Appl. Opt.* **35** 1485, 1996c.
- Lucek, J.K., P. Gunning, D.G. Moodie, K. Smith and D. Pitcher. *Electron. Lett.* **33** 887, 1997.
- Manning, R.J. and D.A.O. Davies. *Opt. Lett.* **19** 889, 1994.
- Manning, R.J., D.A.O. Davies, D. Cotter and J.K. Lucek. *Electron. Lett.* **30** 787, 1994.
- Manning, R.J., A.D. Ellis, A.J. Poustie and K.J. Blow. *J. Opt. Soc. Am. B* **14** 3204, 1997.
- Mark, J. and J. Mork. *Appl. Phys. Lett.* **61** 2281, 1992.
- Mathason, B.K., H. Shi, I. Nitta, G.A. Alphonse, J. Abeles, J.C. Connolly and P.J. Delfyett. *IEEE Photon. Technol. Lett.* **11** 331, 1999.
- Mikkelsen, B., M. Vaa, N. Storkfelt, T. Durhuus, C. Joergensen, R.J.S. Pedersen, S.L. Danielsen, K.E. Stubkjaer, M. Gustavsson and W. Van Berlo. *OFC'95*, San Diego, CA, 1995.
- Nakamura, S. and K. Tajima. *Jpn. J. Appl. Phys.* **35** L1426, 1996.
- Nakamura, S., Y. Ueno and K. Tajima. *IEEE Photon. Technol. Lett.* **10** 1575, 1998.
- Nelson, B.P. and N.J. Doran. *Electron. Lett.* **27** 204, 1991.
- Neogi, A., O. Wada, Y. Takahashi and H. Kawaguchi. *Opt. Lett.* **23** 1212, 1998.
- Nielsen, T.N., A.J. Stentz, K. Rottwitz, D.S. Vengsarkar, Z.J. Chen, P.B. Hansen *et al.* *OFC'00 Post-Deadline Papers*, Baltimore, MD, 2000.
- Nowatzky, A. and P.R. Prucnal. *22nd International Symposium on Computer Architecture*, Santa Margherita, Ligure, Italy, 1995.
- O'Neill, A.W. and R.P. Webb. *Electron. Lett.* **26** 2008, 1990.
- Optical Delay Generator FR-203, Femtochrome Research, Berkeley, CA.
- Otsuka, K. *Opt. Lett.* **8** 471, 1983.
- Patel, N.S., K.L. Hall and K.A. Rauschenbach. *Opt. Lett.* **21** 1466, 1996.
- Patel, N.S., K.L. Hall and K.A. Rauschenbach. *Appl. Opt.* **37** 2831, 1998.
- Perrier, A. and P.R. Prucnal. *J. Lightwav. Technol.* **7** 983, 1989.
- Runser, R.J., K.-L. Deng, P. Toliver, I. Glesk and P.R. Prucnal. *Electron. Lett.* **35** 1097, 1999.
- Runser, R.J., C. Coldwell, P. Toliver, I. Glesk and P.R. Prucnal. *LEOS'00*, Rio Grande, Puerto Rico, 2000.
- Sano, A., Y. Miyamoto, T. Kataoka and K. Hagimoto. *J. Lightwav. Technol.* **16** 977, 1998.
- Seo, S.W., B.Y. Yu and P.R. Prucnal. *Appl. Opt.* **36** 3142, 1997.
- Sokoloff, J.P., P.R. Prucnal, I. Glesk and M. Kane. *OSA Proc. on Photonics in Switching*, (Washington, DC, 1993), PD-4 **16** 1993. Sokoloff, J.P., P.R. Prucnal, I. Glesk and M. Kane. *IEEE Photon. Technol. Lett.* **5** 787, 1993.
- Stone, H.S. *IEEE Trans. Comp.* **C-20** 153, 1971.
- Suzuki, K., K. Iwatsuki, S. Nishi and M. Saruwatari. *Electron. Lett.* **30** 1501, 1994.
- Tajima, K., S. Nakamura, Y. Ueno, J. Sasaki, T. Sugimoto, T. Kato, T. Shimoda, H. Hatakeyama, T. Tamanuki and T. Sasaki. *IEICE Trans. Electron.* **E83-C** 959, 2000.
- Tang, J.M. and K.A. Shore. *IEEE J. Quantum Electron.* **34** 1263, 1998.
- Toliver, P., I. Glesk, R.J. Runser, K.-L. Deng, B.Y. Yu and P.R. Prucnal. *J. Lightwav. Technol.* **16** 2169, 1998.
- Toliver, P., K.-L. Deng, I. Glesk and P.R. Prucnal. *IEEE Photon. Technol. Lett.* **11** 1183, 1999.
- Toliver, P., R.J. Runser, I. Glesk and P.R. Prucnal. *Opt. Commun.* **175** 365, 2000a.
- Toliver, P., I. Glesk and P.R. Prucnal. *Opt. Commun.* **173** 101, 2000b.
- Tsukada, M., W.D. Zhong, T. Matsunaga, M. Asobe and T. Oohara. *J. Lightwav. Technol.* **14** 1979, 1996.
- Wada, O. *Opt. Quantum Electron.* **32** 453, 2000.
- Wolfson, D., A. Kloch, T. Fjelde, C. Janz, B. Dagens and M. Renaud. *IEEE Photon. Technol. Lett.* **12** 332, 2000.
- Yamada, E., K. Suzuki and M. Nakazawa. *Electron. Lett.* **30** 1966, 1994.

R.J. RUNSER *ET AL.*

Yu, B.Y., P. Toliver, R.J. Runser, K.-L. Deng, D. Zhou, I. Glesk and P.R. Prucnal. *IEEE Micro* **18** 28, 1998.

Zhou, D., Hybrid optical TDM/WDM network technology and architecture: component and subsystem development. In *Electrical Engineering Dept.*, Princeton University: Princeton, NJ, 2000.

Zhou, D.Y., K.I. Kang, I. Glesk and P.R. Prucnal. *J. Lightwav. Technol.* **17** 298, 1999.





ELSEVIER

1 January 2000

OPTICS  
COMMUNICATIONS

Optics Communications 173 (2000) 101–106

www.elsevier.com/locate/optcom

# All-optical clock and data separation technique for asynchronous packet-switched optical time-division-multiplexed networks

Paul Toliver<sup>\*</sup>, Ivan Glesk, Paul R. Prucnal

*Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA*

Received 22 July 1999; accepted 17 September 1999

## Abstract

We propose and experimentally demonstrate an all-optical technique for separating a clock synchronization pulse from an optical time-division-multiplexed (OTDM) 100 Gb/s data packet. The technique is based on an all-optical switching device combined with optical feedback. This approach removes limitations found in other techniques such as those that are sensitive to long strings of zeroes in the data packet. © 2000 Elsevier Science B.V. All rights reserved.

**Keywords:** OTDM; Packet switching; Synchronization

## 1. Introduction

One of the key challenges to be solved in photonic packet-switched networks based on optical time-division multiplexing (OTDM) is the synchronization of the ultra-high-speed packets with the control processing at the network nodes. This is necessary not only for the extraction of the packet routing bits in the packet header as illustrated in Fig. 1 but also for the detection of the packet payload once it finally reaches its destination node. Generally, this synchronization can be accomplished for asynchronous packet-switched networks by sending at least one optical clock synchronization pulse with each packet transmitted. The techniques for multi-

plexing the clock pulse with each packet can be divided into five categories including space, wavelength, polarization, amplitude, and time-division approaches. Note that we will not discuss synchronous clock recovery techniques such as phase lock loops, injection locking and other approaches that are applicable only to synchronous optical TDM networks.

Space-division multiplexing of the clock pulse and data packet is the easiest to implement. The clock is simply carried on a separate transmission fiber from the data packets. Of course, there are a number of practical implementation issues that complicate such a design. For example, environmental effects such as slow temperature variations may effect the fibers unequally, thereby causing a time-varying differential delay between the clock and data, which is difficult to compensate. The use of blown fiber, which is present in some existing installations, can minimize these types of effects [1], but

<sup>\*</sup> Corresponding author. Tel.: +1-609-258-2041; fax: +1-609-258-2158; e-mail: ptoliver@ee.princeton.edu

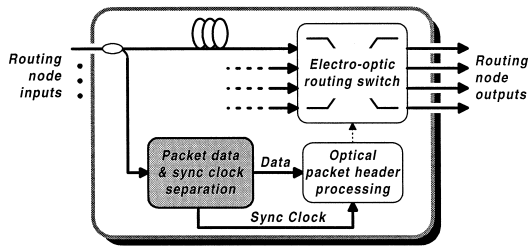


Fig. 1. Conceptual block diagram of optical TDM network routing node architecture including subsystem for all-optical packet data and clock separation.

the cost of installing a separate clock fiber for each network node in new installations may be prohibitive or even impractical for wide area networks. Wavelength-division multiplexing of clock and data has been used in some research testbeds based upon optical TDM [2]. However, due to the effects of chromatic dispersion, this approach is only practical for single-hop networks such as point-to-point links or broadcast-and-select star networks where the path lengths between nodes can be pre-determined. It can not be applied easily to asynchronous packet-switched networks since the optical path length through which a packet travels is non-deterministic. Therefore, the relative delay between the clock and data will also be random. Some of the highest capacity research demonstrations of packet-switched optical TDM have used an orthogonal polarization alignment of the clock and data for system synchronization [3]. This approach becomes limited in larger networks since it is difficult to maintain the correct polarization throughout the system due to polarization mode dispersion (PMD) and other nonlinear effects. Another synchronization approach is to use a higher intensity for the optical clock pulse to differentiate it from the data [4]. In this approach, however, the clock pulse amplitude and its position become difficult to maintain in long distance transmission due to the larger impact of fiber nonlinearities.

The last synchronization technique, time-division multiplexing of clock and data, is used as the basis for the approach taken in this work. Two other time-division approaches have been explored recently by others. The first technique uses fractional bit spacing in order to derive a sync pulse [5]. The

second technique is a self-synchronization approach that positions the clock pulse at the head of the packet in time and uses a fast saturated, slowly recovery gain medium followed by an intensity discriminator to recover the clock [6]. However, this technique suffers from the occurrence of false synchronization pulses created with data packets having long strings of zeroes followed by a one. Also, in both of these time-division approaches, although synchronization is achieved, the clock pulse is not completely separated from the data packet. For the multi-hop routing protocol we have proposed [7], not only is it important to generate a reference clock pulse for synchronization, but it is also necessary to separate the clock pulse completely from the data packet. This is required so that the clock can be repositioned with respect to the data packet for the next hop in the network. In an attempt to solve some of these issues for multi-hop optical TDM networks, we are proposing the use of the all-optical clock and data extraction unit described in the next section.

## 2. Clock and data extraction device

The proposed clock and data extraction unit, illustrated in Fig. 2, is composed of a semiconductor-based nonlinear interferometer, such as the terahertz optical asymmetric demultiplexer (TOAD) [8], combined with optical feedback [9]. The TOAD device, as shown in the schematic of Fig. 2, consists of an optical loop mirror combined with a semiconductor optical amplifier (SOA) offset from the center of the loop by  $\Delta x$ . Data pulses entering the TOAD are split at the 50:50 coupler into two counter-propagating pulses that travel in opposite directions around the loop. In the absence of a clock pulse and for relatively low data pulse energies compared to the SOA saturation energy, the two data pulses experience the same relative phase shift. When they recombine at the 50:50 coupler, the interference condition is such that they are reflected back to the input of the device. If instead a high energy clock pulse is coupled into the loop, the asymmetry of the SOA within the loop creates a differential refractive index, and therefore a differential phase shift, that is seen by the two counter-propagating pulses. If the SOA parameters and clock energies are chosen so that the relative

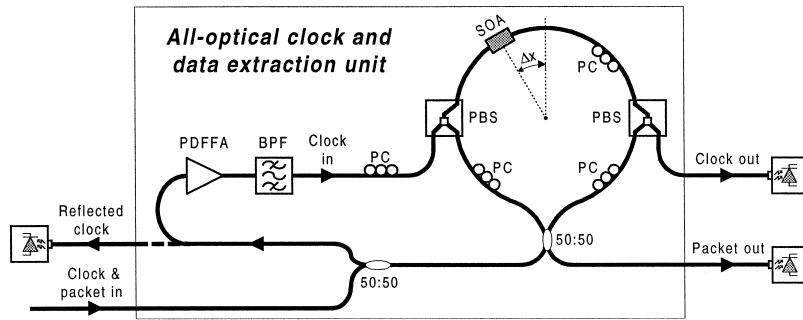


Fig. 2. All-optical clock and data extraction unit. BPF: bandpass filter, PC: polarization controller, PBS: polarizing beam splitter, PDFFA: praseodymium-doped fluoride fiber amplifier, SOA: semiconductor optical amplifier.

phase shift between the counter-propagating pulses is equal to  $\pi$ , the pulse can be transmitted to the output of the device.

Therefore, the TOAD is capable of creating fast switching windows in time for the processing of ultrafast optical pulse streams. The size of this switching window is determined by the offset  $\Delta x$  of the SOA from the center of the loop and is equal to  $2\Delta x/c_f$ , where  $c_f$  is the speed of light in fiber. By making the SOA offset small, a single pulse can be demultiplexed from an ultrafast packet. By making the offset large an entire packet can be switched. In order to describe the principle of our proposed clock and data separation technique, refer to the timing diagram shown in Fig. 3. Each packet of data is preceded by an equal amplitude clock pulse that is spaced  $t_{cp}$  ahead of the packet in time and can have the same polarization, wavelength, and pulse amplitude as the optical data packet pulses. This sync pulse is the first to enter the TOAD device, and since

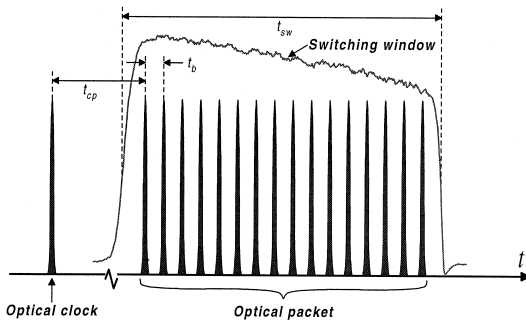


Fig. 3. Timing diagram for illustrating the proposed clock and data separation technique.

no clock pulse is present within the loop, the pulse is reflected from the loop. By feeding the reflected pulse back to the clock port of the TOAD and amplifying it to a higher pulse energy, it can be used to open a wide switching window to let the remainder of the packet data pass to the data output of the TOAD. The temporal width of the switching window,  $t_{sw}$ , must be large enough to switch the entire packet as shown in Fig. 3. Therefore, with this approach, the synchronization clock pulse is sent to the clock output port while the data packet (with the clock pulse completely removed) is sent to the data output port. By using a wide optical gating window to switch the entire data packet, this clock and data separation technique is insensitive to long strings of zeroes in the data packet.

### 3. Experimental setup and results

To demonstrate the principle of our proposed clock and data separation technique, we have constructed the setup illustrated in Fig. 2. A  $1.3 \mu\text{m}$  solid-state laser was used as a source of 2 ps optical pulses that were generated at a 100 MHz rate. An optical packet compression device [10] enabled us to create arbitrary 16-bit data packets with only 10 ps bit spacing or effectively a 100 Gb/s packet bit rate. The TOAD was constructed using a 500  $\mu\text{m}$  long SOA having a bias current of 40 mA. Polarizing beam splitters (PBS) were used to couple the clock pulse into and out of the loop. Pulses reflected from the TOAD were amplified by a praseodymium-doped fluoride fiber amplifier (PDFFA) and filtered by a 3

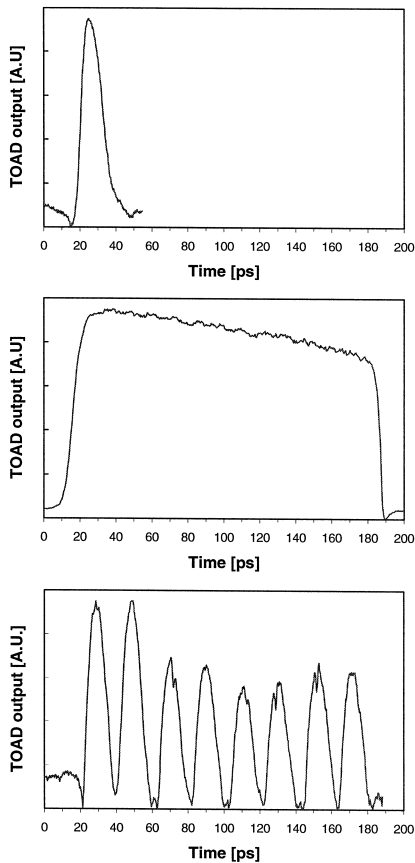


Fig. 4. (Top panel) Scan of narrow TOAD switching window (FWHM = 12 ps) taken with SOA close to the loop center; (middle panel) scan of wide TOAD switching window for gating an entire 16-bit 100 Gb/s OTDM packet (FWHM = 165 ps); and (lower panel) scan of a 16-bit 100 Gb/s packet using the TOAD with a narrow switching window. The packet was encoded with alternating ones and zeroes.

nm optical bandpass filter (BPF) before being fed back to the clock input port of the TOAD at an orthogonal polarization to the data packet. Both the clock and data packet outputs of the TOAD were monitored with relatively low-bandwidth (125 MHz) photodetectors.

The size of the TOAD switching window can be easily changed by setting the appropriate offset of the SOA from the center of the fiber loop. Depending upon the application, the window can be made narrow for demultiplexing a single packet bit or can be made wider for switching an entire packet. We characterized its profile for two different SOA offsets using a time-domain spectroscopy approach by

sending a data pulse into the data input port and positioning a clock pulse relative to the data pulse using an optical delay stage. The output of the TOAD is then recorded with respect to the clock position in time. Fig. 4(top panel) shows the experimental results of the window scan for the case when the SOA is close to the center of the fiber loop resulting in a full width half maximum (FWHM) window size of 12 ps. The SOA offset was then increased to create a switching window having a width of approximately 165 ps as shown in Fig. 4(middle panel). The slow decrease in the switching window amplitude as a function of time is due to carrier-recovery dynamics occurring within the SOA [8]. Finally, using the TOAD with the narrow switching window, we were able to measure a 16-bit, 100 Gb/s data packet as illustrated in Fig. 4(lower panel). Note that although this packet is encoded with alternating ones and zeroes, the packet compression device is capable of creating any arbitrary 16-bit, 100 Gb/s packet. Also note that the wide TOAD switching window shown in Fig. 4(lower panel) is large enough to switch this entire packet of data. Finally, the bit amplitude fluctuations seen in the packet scan are due primarily to the unequal power splitting ratios of the couplers used in the packet compression device.

To illustrate the operation of the clock and data extraction unit, a sequence of four packets were injected into the device. As illustrated in Fig. 5, a clock synchronization pulse precedes each data packet. Because of the length involved in the feedback path, this pulse was placed  $t_{cp} = 160$  ns ahead of the packet. Each packet of data contains only a

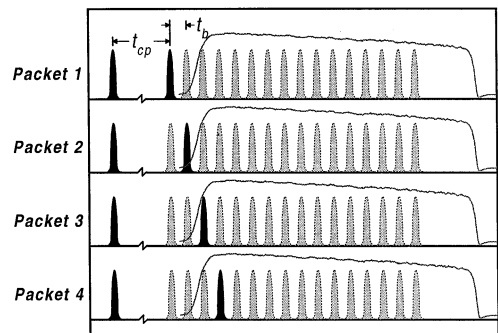


Fig. 5. Sequence of four packets and corresponding clock pulses used to illustrate operation of the clock and data separation unit.

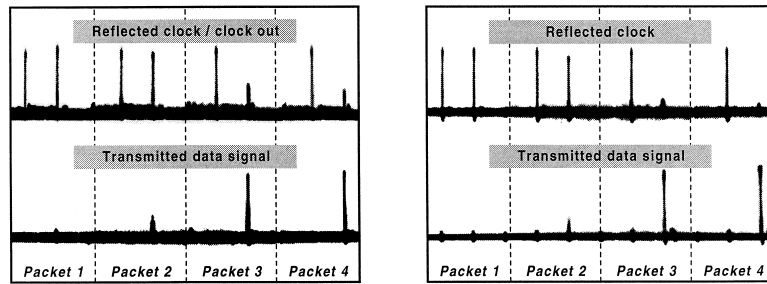


Fig. 6. (Left panel) Reflected (top) and transmitted (bottom) signal measured at the TOAD for the sequence of four packets illustrated in Fig. 5 using the self-clocking mode of operation with optical feedback; and (right panel) same as (left panel) but using a separate clock signal for packet data extraction and without optical feedback.

single 1-bit to make the switching operation clear, but each bit is shifted in time to the next position ( $t_b = 10$  ps) in each of the packets as shown in Fig. 5. In order to distinguish between pulses that are transmitted or reflected from the TOAD, the switching window is aligned so that only the last 14-bits of the packet are gated.

Fig. 6 shows the experimental results for the sequence of four packets shown in Fig. 5. The top waveform is the reflected signal from the TOAD measured at the clock output whereas the bottom waveform is the transmitted output measured at the data output. The first pulse seen in the reflected port is the initial clock synchronization pulse for the packet. Since the data pulse in the first packet falls outside the TOAD switching window, it is also reflected and seen at this port. Note also that there is no transmitted signal at this moment in time. The same sequence occurs for the second packet, but this time, there is a small amount of pulse leakage seen at the transmitted port due to the finite rising edge of the TOAD switching window (see Fig. 5). As shown in Fig. 5, the data pulses in the third and fourth packet fall within the switching window and should be switched out. As seen in the last two packet sequences in Fig. 6(left panel), this is what occurs in the experiment. First, the clock pulse is reflected and next, the data pulse within the packet is sent to the packet output. Note, in this case however, there is some data pulse leakage seen at the reflected port due to incomplete TOAD switching. The 12 dB signal gain of the PDFFA used in the experiment is insufficient to amplify the reflected pulse to induce enough phase shift to completely switch the packet

data pulse to the transmitted port. This was verified by using a separate clock pulse and removing the feedback path. As shown in Fig. 6(right panel), this clock pulse had sufficient energy (500 fJ) to completely switch the data pulse and the reflected signal was suppressed. Therefore, by using a higher gain semiconductor optical amplifier combined with optical integration techniques, we believe the clock and data separation unit can be practical for use in asynchronous packet-switched OTDM networks with cascaded all-optical switches [11].

#### 4. Conclusion

We have proposed a robust clock and data separation technique for asynchronous packet-switched networks based on an all-optical switching device combined with optical feedback. In comparison to other proposed techniques, this approach uses only a single clock pulse separated in time by a fixed amount from the data packet, but the pulse has the same wavelength and amplitude as pulses in the data packet. In the experimental demonstration, this separation in time was limited to 160 ns due to the use of bulk optical components. The use of semiconductor optical amplifiers combined with optical integration technology can reduce this to a small fraction of the packet duration. The device provided good suppression of the clock from the packet signal although the clock signal contained some data signal leakage that would limit the cascability of an all-optical packet switch configuration. We showed this leakage is reduced, however, by using a clock pulse with suffi-

cient energy (only 500 fJ). Since this technique is self-synchronizing, it allows for self-routing capabilities in ultrafast asynchronous packet-switched networks based on optical TDM.

## References

- [1] P. Gunning, J.K. Lucek, D.G. Moodie, K. Smith, D. Pitcher, Q. Badat, A.S. Siddiqui, *Electron. Lett.* 34 (1998) 488.
- [2] Y. Shimazu, M. Tsukada, *J. Lightwave Technol.* 10 (1992) 265.
- [3] I. Glesk, J.P. Sokoloff, P.R. Prucnal, *Electron. Lett.* 30 (1994) 1322.
- [4] K.-L. Deng, I. Glesk, K.I. Kang, P.R. Prucnal, *IEEE Photon. Technol. Lett.* 9 (1997) 830.
- [5] M. Shabeer, J.K. Lucek, K. Smith, D. Cotter, D.C. Rogers, *Electron. Lett.* 31 (1995) 1476.
- [6] T.J. Xia, Y.H. Kao, Y. Liang, J.W. Lou, K.H. Ahn, O. Boyraz, G.A. Nowak, A.A. Said, M.N. Islam, *IEEE Photon. Technol. Lett.* 11 (1999) 269.
- [7] S.W. Seo, B.Y. Yu, P.R. Prucnal, *Appl. Opt.* 36 (1997) 3142.
- [8] J.P. Sokoloff, P.R. Prucnal, I. Glesk, M. Kane, *IEEE Photon. Technol. Lett.* 5 (1993) 787.
- [9] K.J. Blow, R.J. Manning, A.J. Poustie, *Opt. Commun.* 134 (1997) 43.
- [10] K.-L. Deng, K.I. Kang, I. Glesk, P.R. Prucnal, *Electron. Lett.* 33 (1997) 1237.
- [11] I. Glesk, K.I. Kang, P.R. Prucnal, *Electron. Lett.* 33 (1997) 794.



ELSEVIER

1 March 2000

OPTICS  
COMMUNICATIONS

Optics Communications 175 (2000) 365–373

[www.elsevier.com/locate/optcom](http://www.elsevier.com/locate/optcom)

# Comparison of three nonlinear interferometric optical switch geometries

Paul Toliver, Robert J. Runser<sup>\*</sup>, Ivan Glesk, Paul R. Prucnal

*Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA*

Received 8 November 1999; received in revised form 28 December 1999; accepted 29 December 1999

## Abstract

We present an experimental study of ultrafast all-optical interferometric switching devices based upon a resonant nonlinearity in a semiconductor optical amplifier (SOA). We experimentally compare three configurations: one based upon a Sagnac interferometer and the other two based upon Mach–Zehnder interferometers. By using picosecond pulses, we characterize the switching window of the three devices in terms of both temporal width and output peak-to-peak amplitude. These results are found to be in close agreement with a previously developed theoretical model. Since these nonlinear interferometric switches use an active device as the nonlinear element, relatively low control pulse energy is needed to perform switching as compared to other techniques. As a result, these optical switches are practical for all-optical demultiplexing and ultrafast optical sampling for future lightwave communication systems. © 2000 Published by Elsevier Science B.V. All rights reserved.

**Keywords:** Optical communications; Semiconductor optical amplifiers; Optical time division multiplexing; Terahertz optical asymmetric demultiplexer

## 1. Introduction

Increasing the bandwidth capacity of lightwave networks has recently received considerable attention from the research community to address the growing traffic demands on today's communication systems. All-optical switches and demultiplexers are an important development that enables high aggregate data rates to be achieved in optical time division multiplexed (OTDM) networks. Semiconductor optical nonlinearities with long recovery times ( $\geq 100$  ps) have been used to demonstrate interferometric all-

optical switches that promise to deliver switching and demultiplexing at terabit/s rates. These nonlinearities are typically based upon a resonant excitation in active or passive semiconductor nonlinear waveguides or optical amplifiers. Extensive experimental [1–5] and theoretical [6–9] analysis has been performed on various interferometric configurations of these devices. Due to the compact nature of these devices, many have been integrated indicating their practicality for future communication systems [10,11]. Optical switches which use an active semiconductor optical amplifier (SOA) as the nonlinear switching element have performed efficient switching demonstrations [12] using low control pulse energy (250 fJ) as compared to passive devices. As a

<sup>\*</sup> Corresponding author. Tel.: +1-609-258-2041; fax: +1-609-258-2158; e-mail: [rrunser@ee.princeton.edu](mailto:rrunser@ee.princeton.edu)

consequence, this paper focuses on the SOA-based interferometric switches.

We present an experimental follow-up study to a previously developed theoretical model [6] for three different interferometric geometries that use a semiconductor resonant nonlinearity to achieve optical switching. The performance of each device is evaluated and compared based upon the temporal response of the optical switch. By varying the switching window temporal size, both the relative output amplitude and minimum achievable window duration can be investigated for each geometry. We find that the experimental results of this study are in close agreement with the predicted values from the theoretical model. Finally, we conclude with a summary of the performance advantages of the three optical switches.

## 2. Optical switch geometries

Fig. 1 shows the three interferometric optical switches that we investigate both experimentally and theoretically. The first geometry, shown in Fig. 1, upper panel, is a Sagnac interferometer with an SOA offset from the center position of the fiber loop and is known as a terahertz optical asymmetric demultiplexer (TOAD) [1]. In the absence of a control signal, data pulses enter the fiber loop, pass through the SOA at different times as they counterpropagate around the loop, and recombine interferometrically at the 50/50 coupler at the base of the loop. Since both pulses see the same medium as they propagate around the loop, the data is reflected back toward the source. In the presence of the control signal, switching can occur. The control signal energy is chosen to be at least ten times the data pulse energy. When the control signal is injected into the loop, it saturates the SOA and changes its index of refraction. As a result, a differential phase shift can be achieved between the two counterpropagating data pulses to switch the data pulses to the output port. A polarization or wavelength filter is used at the output to reject the control signal and pass the switched data signal. Since the SOA is slowly recovering, data pulses that enter the switch immediately after the control pulse both see the SOA in approximately the

same recovery state and do not experience a significant differential phase shift. The temporal duration of the switching window is determined by the offset of the SOA,  $\Delta x$ , from the center position of the loop. As this offset is reduced, the switching window size decreases. The size of the nominal switching window duration,  $\tau_{\text{win}}$ , is related to the offset position by  $\tau_{\text{win}} = 2\Delta x/c_{\text{fiber}}$  (where  $c_{\text{fiber}}$  is the speed of light in fiber).

The other two switch geometries shown in Fig. 1, middle panel, and Fig. 1, lower panel, are both based upon a Mach–Zehnder interferometer. In the absence of the control signals, the Mach–Zehnder is balanced in such a way as to send all data signals entering the device to the reject port (not shown on the figure). When control pulses are injected into the interferometer, a differential phase shift is briefly introduced between the two arms of the interferometer causing a data pulse to be switched to the output port. Similar to the TOAD, subsequent data pulses that pass through the switch see the slow recovery of both SOAs and are rejected. The slight difference between the two Mach–Zehnder geometries shown is with respect to the propagation direction of control and data signals. In the colliding pulse Mach–Zehnder (CPMZ) shown in Fig. 1, middle panel, the data and control signals counter-propagate through the interferometer. As a result, a filter is not needed at the output port to reject the control signals. However, the symmetric Mach–Zehnder (SMZ) in Fig. 1, lower panel, requires a filter at the output port to reject the control signals from the switched data signal since data and control signals co-propagate through the interferometer. Assuming the SOAs are positioned in the same relative location within the interferometer, the nominal switching window for both Mach–Zehnder configurations is determined by the temporal control pulse separation,  $\Delta t_{\text{cs}}$ , of the signals Control 1 and Control 2 prior to entering the interferometer such that  $\tau_{\text{win}} = \Delta t_{\text{cs}}$ .

Although the nominal switching window size provides an estimate of the switching window temporal duration, it does not account for the finite length of the SOAs within the interferometer. While the SOA length has little effect on the SMZ geometry, the minimum achievable switching windows for both the TOAD and CPMZ are constrained by the length of the SOAs [6]. By incrementally reducing the switch-



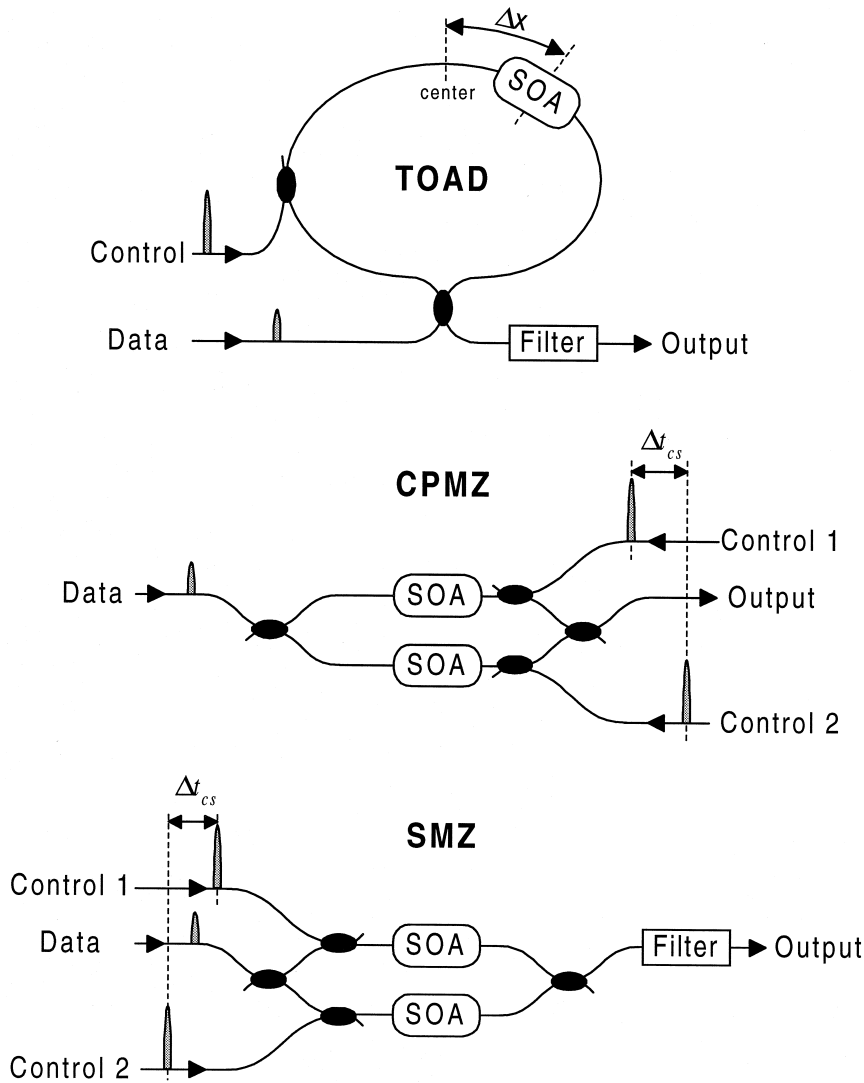


Fig. 1. All-optical SOA-based interferometric switch geometries: upper panel: terahertz optical asymmetric demultiplexer (TOAD), middle panel: colliding-pulse Mach–Zehnder (CPMZ), and lower panel: symmetric Mach–Zehnder (SMZ).

ing window temporal duration of all three geometries, we explore the effect of the SOA length on the temporal response of the switches both theoretically and experimentally in Sections 3 and 4.

### 3. Nonlinear interferometer model

In this section, we develop a theoretical model to describe the operation of the three optical switch

geometries. This model, which is based on previous work [6], is used in Section 4 to analyze the impact of the effective switching offset on the optical switching window performance. Note that although this model is not a rigorous device-level model, it does provide relatively good agreement between the predicted optical switch performance and experimental data. Since the switch geometries discussed in this paper are all based upon optical interferometry,

the output signal of the switch can be described by the following basic interferometric equation:

$$P_{\text{out}}(t) = \frac{P_{\text{in}}(t)}{4} \left\{ G_1(t) + G_2(t) - 2\sqrt{G_1(t)G_2(t)} \cos(\phi_1(t) - \phi_2(t)) \right\}. \quad (1)$$

The input data signal is represented by  $P_{\text{in}}(t)$ , and it is assumed that the interferometer is balanced so that there is initially no signal at the output. In the presence of one or more control signals, the two data signals (signals 1 and 2) that interfere within the interferometer experience a time-dependent gain,  $G_{1,2}(t)$ , and phase shift,  $\phi_{1,2}(t)$ , as they traverse the SOA. By introducing some form of asymmetry within the interferometer, either by appropriate timing of the control pulse(s) or by careful positioning of the nonlinear element(s), the two signals can experience different responses. This differential response results in the input data signal being switched to the output of the device.

For the simple model adopted here, the gain and phase shift experienced by signals travelling through the SOA is assumed to be described by the following temporal responses:

$$G_{1,2}(t) = G_0 - \Delta G \int_{-\infty}^{\infty} h_{1,2}(t') P_{\text{clk}}(t' - t) dt' \quad (2)$$

$$\phi_{1,2}(t) = \Delta \phi \int_{-\infty}^{\infty} h_{1,2}(t') P_{\text{clk}}(t' - t) dt'. \quad (3)$$

The variable  $P_{\text{clk}}(t)$  represents the control signal whereas  $h(t)$  represents the impulse response of the SOA. Also,  $G_0$  is the initial gain of the SOA and  $\Delta G$  and  $\Delta \phi$  represent the amount of change in gain or phase when the control pulse traverses the SOA. Depending upon the direction of propagation of the data signal with respect to the control pulse, different impulse response functions must be used in Eqs. (2) and (3). For the case when the data and control signals co-propagate, the following single time-constant impulse response is assumed:

$$h_{\text{co}}(t) = \Phi(t) \exp\left(\frac{-t}{\tau_{\text{SOA}}}\right). \quad (4)$$

The step function,  $\Phi(t)$ , accounts for the situation when the data signal is either leading or trailing the

control pulse and  $\tau_{\text{SOA}}$  is assumed to be the dominant SOA recovery time constant. In contrast, the length of the SOA must be taken into account for the case when the data and control signals counter-propagate with respect to each other. The impulse response for the counter-propagating geometry is given as:

$$h_{\text{cntr}}(t) = \frac{\exp(-t/\tau_{\text{SOA}})}{l} \int_{-l/2}^{l/2} \Phi\left(x + \frac{c_{\text{SOA}}t}{2}\right) \times \exp\left(\frac{-2x}{c_{\text{SOA}}\tau_{\text{SOA}}}\right) dx. \quad (5)$$

In this equation, the step function accounts for the intersection of the counter-propagating data and control signals. Finally,  $c_{\text{SOA}}$  is the speed of the light through the SOA and  $l$  is its length.

Using Eqs. (1)–(5), the switching windows of the three optical switch geometries can be computed using the following approach. Based upon the switch geometry, the appropriate impulse responses,  $h_{1,2}(t)$ , are computed for each of the data signals within the interferometer. For the SMZ configuration,  $h_{\text{co}}(t)$  is used for both impulse responses; for the TOAD,  $h_{\text{co}}(t)$  is used for one response and  $h_{\text{cntr}}$  is used for the other; and finally, for the CPMZ,  $h_{\text{cntr}}(t)$  is used for both responses. The selected impulse responses are delayed from one another by the effective switching offset. Control pulses are convolved with each response and the result is used to compute the differential gain and phase evolution described by Eqs. (2) and (3). The output of the interferometer switch is then calculated using Eq. (1). Finally, in order to take into account the finite temporal width of the data signal, the data pulse is correlated with the output response of the interferometer. The resulting signal is normalized to the maximum output in order to simplify the comparisons between the different geometries as well as the experimental data. As in previous work, the constants used in computing the switching window performance are as follows:  $G_0 = 10$ ,  $\Delta G = 5$ ,  $\Delta \phi = 0.5\pi$ ,  $\tau_{\text{SOA}} = 400$  ps,  $l = 500$   $\mu\text{m}$ , and finally  $n_{\text{SOA}} = 3.3$ . Pulses with Gaussian temporal envelopes having a width of 1.6 ps are assumed for both the data and control signals. A comparison of the switching window performance for the three switch geometries using both the theo-

retical model developed here as well as experimental techniques is provided in the next section.

#### 4. Optical switch characterization

To experimentally evaluate the performance of the three nonlinear interferometric optical switches, fiber-based versions of each configuration were constructed from discrete components. The nonlinear element used in each switch geometry is an Alcatel 1901 SOA biased at approximately 100 mA. Polarization controllers are used internally within each switch in order to align the interferometer for the proper condition for interference.

Each configuration was characterized using the test setup shown in Fig. 2. A 1.55  $\mu\text{m}$  mode-locked fiber laser (MLL) is used to generate a continuous stream of 1.6 ps optical pulses at a 2.5 GHz rate. The pulse stream is amplified by an erbium-doped fiber amplifier (EDFA) and optically split into control signals and data signals for injection into the optical switch device-under-test (DUT). For the SMZ and CPMZ switch geometries (which require two control pulses) an optical delay line (ODL) is used to set the relative offset between the pulses to control the desired switching window size. In the TOAD geometry, an optical delay line inside the interferometer is used to change the SOA offset and set the switching window. For the SMZ and TOAD configurations, it is necessary to set orthogonal polarization states for

the input control and data pulses using polarization controllers (PC). This enables the separation of the switched data signal from the control pulses at the output of the switch using a polarization filter.

For each switch under evaluation, the data and control pulse energies (average powers) are fixed at 4 fJ (10  $\mu\text{W}$ ) and 50 fJ (125  $\mu\text{W}$ ), respectively, when measured at the input facet of the SOAs inside the interferometer. A mechanical vibrator is used to quickly scan the data pulses in time over a 40 ps range to map out the switching window of the device under test. This technique provides a means of rapidly characterizing the switching window. While the TOAD is based upon the inherently stable Sagnac interferometer, thermal variations in the optical fiber cause the output of the fiber-based Mach–Zehnder switches to fluctuate slowly in time. By performing the scan at a rate faster than the thermal variations, switching windows of the fiber-based Mach–Zehnder switches can be obtained without resorting to complex stabilization techniques. (Note that thermal variations do not significantly affect the stability of any of these interferometers if integrated devices with short optical path lengths are used.)

The results of the TOAD optical switch characterization are described first. Using the test setup, experimental scans of the TOAD switching window were taken for various offsets and the results are summarized in Fig. 3, upper panel. The SOA position within the loop is varied so that the effective switching offset decreases from approximately 18 ps

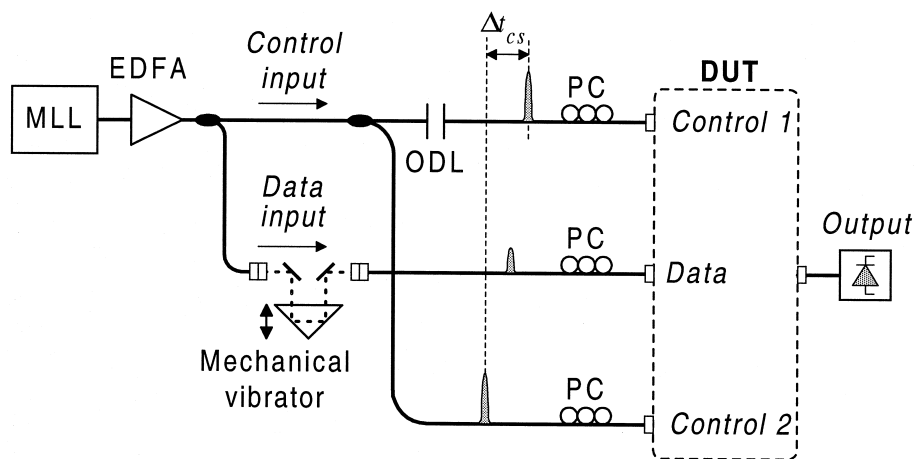


Fig. 2. Experimental test setup for characterizing optical switches.

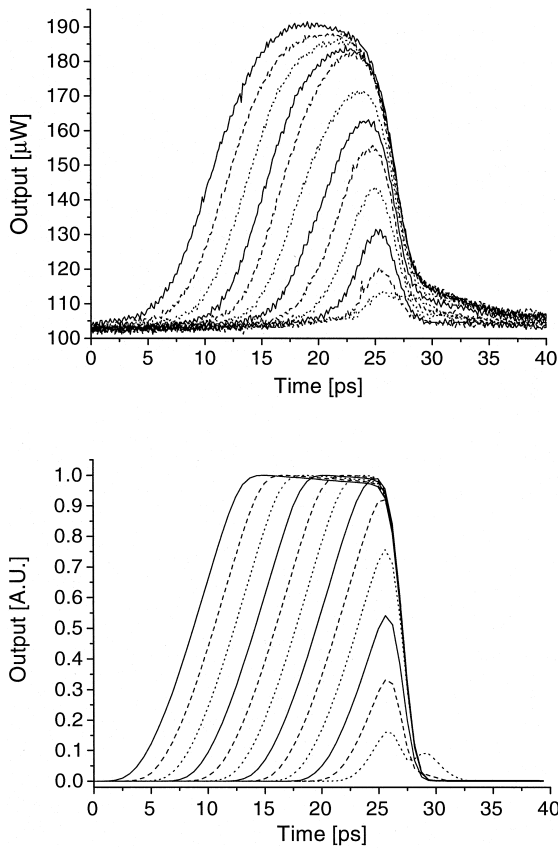


Fig. 3. Upper panel: experimental window scans of the TOAD optical switch, and lower panel: computed window scans using the model described in the text.

down to 0 ps in 1.6 ps steps. As the switching offset is initially decreased from a large offset, the window width decreases by the same proportion although the output amplitude remains relatively constant. At an offset of approximately 10 ps, however, the amplitude begins to decrease in direct proportion as well. This amplitude decrease is a result of the finite length of the SOA becoming a factor. At extremely small offsets, the switching window width does not decrease further since the finite data and control pulse widths become the dominant limitation. This trend continues until the effective switching offset is 0 ps at which point the switching window nearly vanishes. The results of the numerical simulation for the TOAD switching window, shown in Fig. 3, lower panel, are in good agreement with the experimental results. The long rising edge of the switching

window is due to the finite length of the SOA whereas the sharp falling edge is only limited by the data and control pulse widths. Also, the window does not entirely disappear at zero effective switching offset as verified by the experimental data. Note, however, that the tops of the experimental windows for large offsets are not exactly flat as predicted in the simulations. This and other differences may be due to a variety of experimental nonidealities including variations in the source pulse energy and detector noise.

Under the same experimental conditions, similar measurements were taken for the SMZ and CPMZ switch geometries. The results of these scans are shown in Fig. 4, upper panel, and Fig. 5, upper panel. Due to its co-propagating nature, the SMZ has the unique characteristic of sharp edges on both the

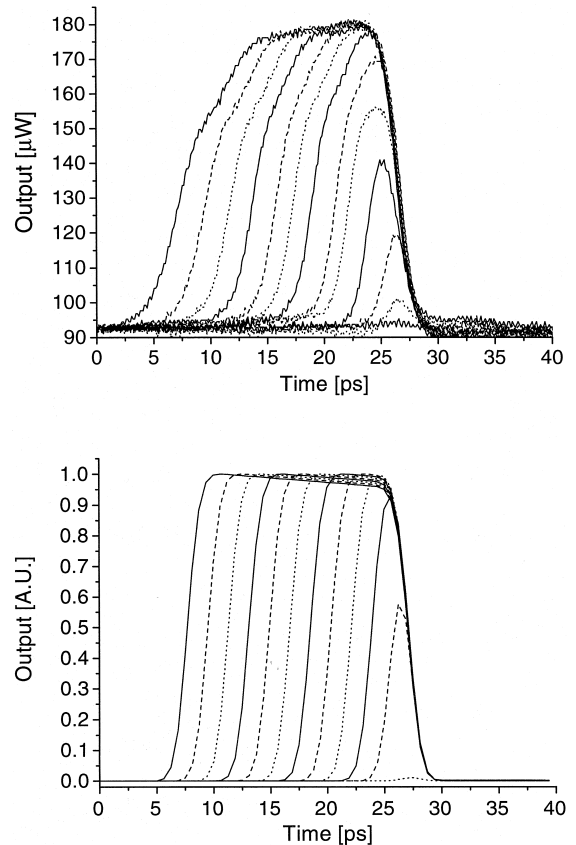


Fig. 4. Upper panel: experimental window scans of the SMZ optical switch, and lower panel: computed window scans using the model.

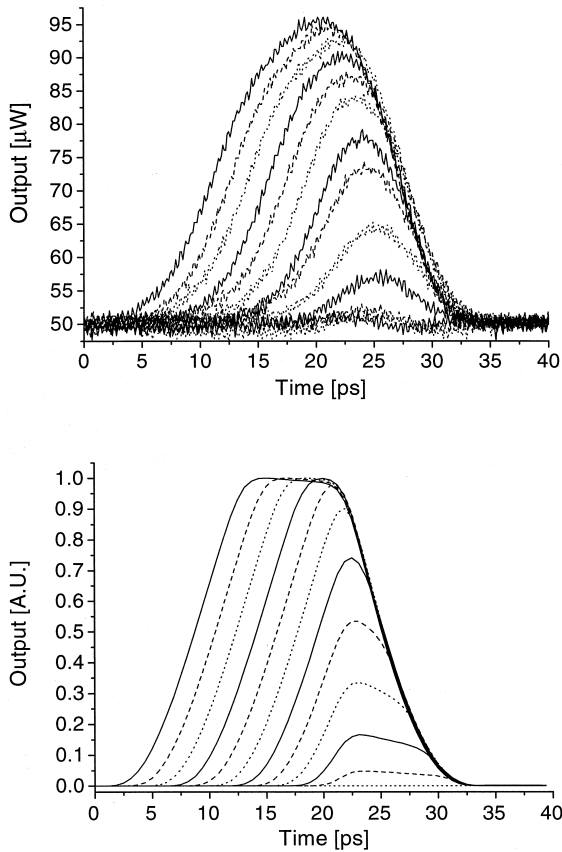


Fig. 5. Upper panel: experimental window scans of the CPMZ optical switch, and lower panel: computed window scans using the model.

rising and falling sides of the switching window. In contrast, since the CPMZ is a purely counter-propagating geometry, both edges rise and fall more slowly due to the finite SOA lengths. The simulated switching windows for the SMZ and CPMZ, shown in Fig. 4, lower panel, and Fig. 5, lower panel, also illustrate these characteristics. There are a few noticeable differences between the simulations and the experimental measurements. Although the tops of the experimental SMZ switching windows are relatively flat for large SOA offsets, the sense of the slope is opposite relative to the simulated values. The change in slope is a result of a 0.6 dB coupling variation throughout the translation length of the mechanical vibrator used to scan the data signal through the switching window. Additionally, there is a change in slope on the leading edge of the experimental win-

dows that does not appear in the simulations. The source of this discrepancy is not fully understood at this time and is under investigation. In order to compare the results of the scans taken for the three optical switch geometries, the window peak-to-peak amplitude and FWHM were computed for each offset. The results of the amplitude measurements on the experimental and theoretical switching windows are summarized in Fig. 6, upper panel, and Fig. 6, lower panel, respectively. As shown in these figures, the CPMZ is the first switch to decrease in amplitude as the effective switching offset decreases. The finite SOA lengths in the counter-propagating geometry cause this reduction in the output amplitude. In contrast, the amplitude of the co-propagating SMZ

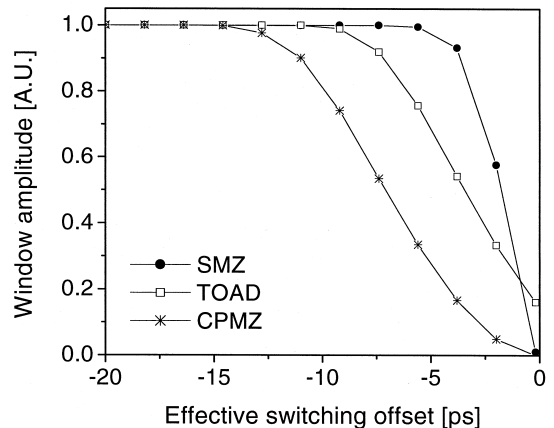
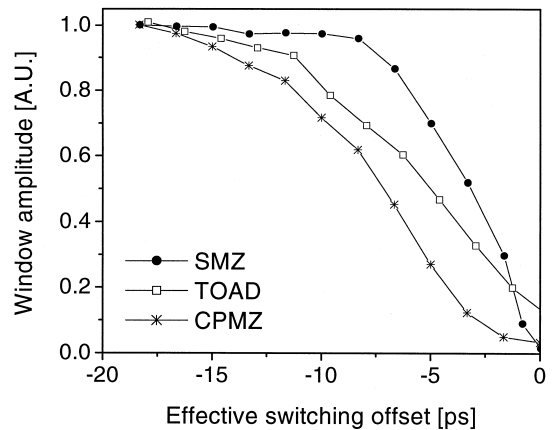


Fig. 6. Upper panel: measured peak-to-peak amplitude of experimental window scans, and lower panel: computed amplitude using the theoretical model.

switch is relatively flat up to very small offsets when the finite pulse widths finally become a factor. Finally, the TOAD geometry is effectively half co-propagating, half counter-propagating so it lies somewhere in the middle. One last interesting note is that while the output amplitude of the SMZ and CPMZ configurations decreases to zero at zero effective switching offset, the output amplitude of the TOAD geometry does not. This can be seen in the simulated results shown in Fig. 6, lower panel, and is also verified experimentally by the data in Fig. 6, upper panel.

Finally, the computed FWHM switching windows for the different geometries are shown in Fig. 7. Fig. 7, upper panel, shows the width of the experimental scans, whereas Fig. 7, lower panel, is computed

using the theoretical model. The minimum detectable switching window sizes measured for the experimental configurations of the SMZ, TOAD, and CPMZ switches are 2.5 ps, 3.5 ps, and 8.3 ps, respectively. These minimum switching window widths agree closely with the minimum values predicted by the theoretical model. When the switching offset is reduced further beyond the values shown in Fig. 7, upper panel, the output amplitude is too small to accurately compute the window width. While the theoretical trend of the window sizes agrees well with the experimental data for the TOAD and SMZ, the experimental data for the CPMZ does not match the simulation as well for small switching offsets. A possible explanation for this difference is that there is greater experimental error in the measurement of the CPMZ window size since the output signal from the CPMZ (see Fig. 5, upper panel) is roughly half that of the TOAD and SMZ and, therefore, closer to the noise floor of the detector. The lower output signal of the CPMZ is due to additional splitting losses experienced by the switched output signal for a given data pulse energy. Furthermore, the simple theoretical model used in this study may not adequately account for other physical device level effects that may influence the switching behavior observed in the experimental trend.

## 5. Conclusion

In conclusion, we have characterized the switching response of three different geometries of ultrafast nonlinear interferometric optical switches. Although there are small deviations, our experimental results show good agreement with a simple impulse response numerical model. Further experimental investigations and refinements of the numerical model to include more detailed physical effects, such as ultrafast semiconductor optical nonlinearities, may account for the differences observed between theory and experiment. Of all the geometries considered, the SMZ switch exhibits the best performance in terms of the minimum switching window width and output peak-to-peak amplitude. The performance of the CPMZ switch, on the other hand, is somewhat limited by the counter-propagating geometry and finite SOA lengths. However, it has the advantage that it is not necessary to reject the control signal at

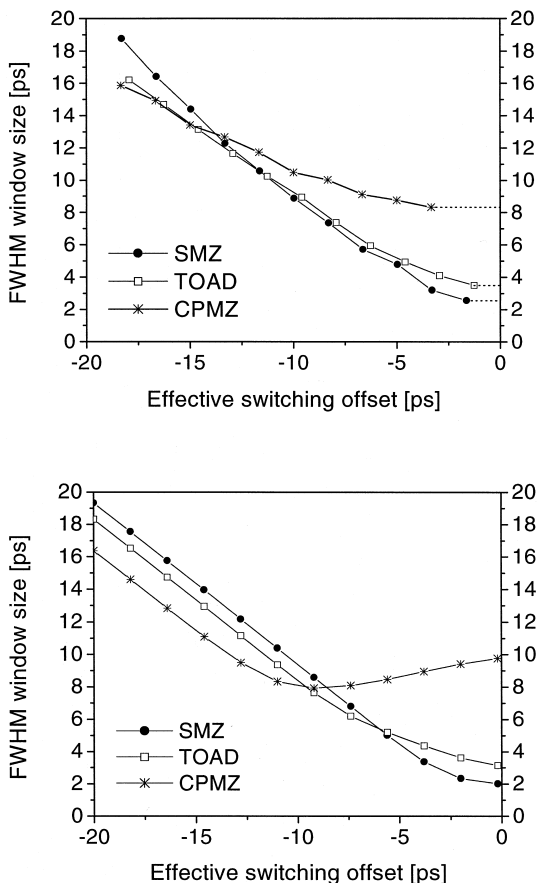


Fig. 7. Upper panel: measured full-width at half-maximum (FWHM) window widths of experimental scans, and lower panel: computed widths using the theoretical model.

the output port. Finally, the TOAD has nearly comparable performance to the SMZ but is an inherently balanced interferometer unlike the two fiber-based Mach–Zehnder geometries used in this study. The control pulse energy requirements of all three devices studied are extremely low ( $< 50$  fJ at the SOA facet) due to the active nature of the nonlinear elements used in the interferometers. This is at least an order of magnitude less than the energy required by passive structures. Although this study only considered fiber-based interferometric switches, these results apply equally to integrated devices based on similar geometries.

## References

- [1] J.P. Sokoloff, P.R. Prucnal, I. Glesk, M. Kane, *IEEE Photon. Technol. Lett.* 5 (1993) 787.
- [2] S. Nakamura, Y. Ueno, K. Tajima, *IEEE Photon. Technol. Lett.* 10 (1998) 1575.
- [3] N.S. Patel, K.L. Hall, K.A. Rauschenbach, *Opt. Lett.* 21 (1996) 1466.
- [4] A.D. Ellis, D.M. Patrick, D. Flannery, R.J. Manning, D.A.O. Davies, D.M. Spirit, *J. Lightwave Technol.* 13 (1995) 761.
- [5] M. Eiselt, W. Pieper, H.G. Weber, *Electron. Lett.* 29 (1993) 1167.
- [6] K.I. Kang, T.G. Chang, I. Glesk, P.R. Prucnal, *Appl. Opt.* 35 (1996) 417.
- [7] R.J. Manning, A.D. Ellis, A.J. Poustie, K.J. Blow, *J. Opt. Soc. Am. B* 14 (1997) 3204.
- [8] N.S. Patel, K.L. Hall, K.A. Rauschenbach, *Appl. Opt.* 37 (1998) 2831.
- [9] S. Nakamura, K. Tajima, *Jpn. J. Appl. Phys.* 35 (1996) L1426.
- [10] J. Leuthold, P.-A. Besse, E. Gamper, M. Dulk, S. Fischer, G. Guekos, H. Melchoir, *J. Lightwave Technol.* 17 (1999) 1056.
- [11] E. Jahn, N. Agrawal, M. Arbert, H.-J. Ehrke, D. Franke, *Electron. Lett.* 31 (1995) 1857.
- [12] K.-L. Deng, R.J. Runser, P. Toliver, C. Coldwell, D. Zhou, I. Glesk, P.R. Prucnal, *Electron. Lett.* 34 (1998) 2418.